Natural Language Processing

Lecture 3
Syntax Analysis - Part 2

Project “ATP” (“AOT”)

http://www.aot.ru/

Fragmentational analysis:

- identifying fragment type;
- applying rules for disambiguation;
- applying rules for hierarchy.

Syntactical analysis: applying rules for forming groups + principle of projectivity

AOT: Identifying fragment type

Fragment types:

- finite verb;
- short participle;
- short adjective;
- predicative word;
- participle;
- gerund;
- infinitive;
- introductory phrase;
- null

Example 1:

на этот раз она не права: short adjective / null

Example 2:

мои права забрали в милиции: finite verb

AOT: Rules for disambiguation

1. If one of the forms is short adjective or short participle and there is not a noun in Nom.case with the same gender and number, then this form is removed: *права он получил только с пятой попытки*;
2. If there is a predication without ambiguity, then in other wordforms homonyms with predication are removed: *мыла на кухне она не нашла*

AOT: Rules for hierarchy

1. ***Написанная в спешке*** *программа выполнила недопустимую операцию*. Is participle before a noun: search for a noun with the same case, gender, number. If a noun is found then this fragment is included in the next right one and the program exits from this rule.
2. *Программа,* ***написанная в спешке****, выполнила недопустимую операцию*. Search for a noun or a pronoun in the same gender, number and case in the left fragment. If a word is found then this fragment is included in the next left one and the program exits from this rule.

AOT: Syntax analysis

Example 1. *рубить дрова*; *есть кашу*; *читать книгу*. Rule: verb + direct object. Chains: verb + noun (Acc. case). Phrase type: verb phrase.

Example 2. *радостно сообщил*; *тихо и смирно сидит*; *хорошо знаю*. Rule: adverb + verb. Chains: adverb + verb, adverb phrase + verb, adverb + verb phrase. Phrase type: verb phrase.

Example 3. *краше тебя*; *уютнее твоего дома*. Rule: forming comparative phrase. Chains: two phrases: the first phrase with an adjective in comparative form as a head, the second phrase with a noun (Gen.case) as a head. Phrase type: adjective phrase.

Link Grammar Parser

https://www.abisource.com/projects/link-grammar/

Stages of analysis:

1. Building a set of all possible syntactical constructions for a sentence. The variants is chosen by the criterion of projectivity and the criterion of minimal coherence;
2. Postprocessing of alternative structures of a sentence.

Link: dictionaries

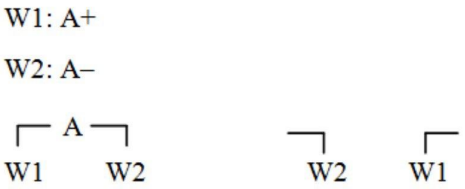
words.n.1	Countable nouns in singular form (*book*)
words.n.2.s	Nouns in plural form ending with “s” (*books*)
words.n.2.x	Nouns in plural form not ending with “s” (*man* - *men*)
words.n.3	Uncountable nouns (*air*)
words.n.4	Nouns which can be both countable and uncountable (*tea*, *coffee*)

Link

Connectors: S (subject - predicate), O (object - predicate) etc.

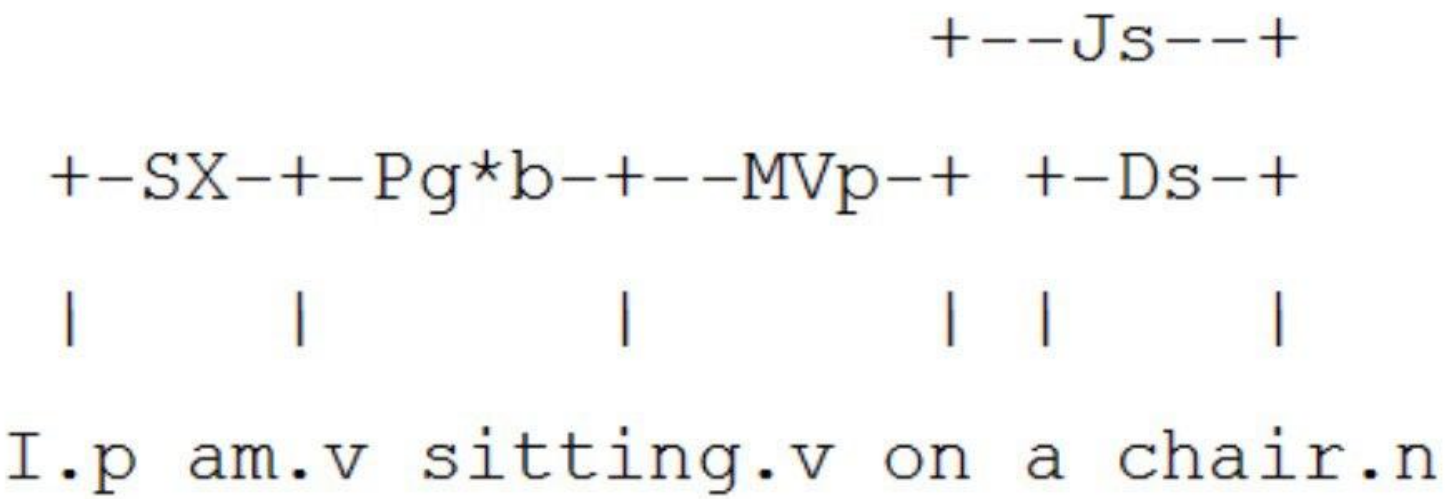
Link directions: “+” - right, “-” - left.

Link = right connector + left connector.



- & - nonsymmetric conjunction. Example: if “*W*: *A*+ & *B*+”, then a word *X*, which has link *A* with word *W*, is located before a word *Y*, which has link *B* with word *W*.
- or - disjunction. If “*W*: *A*+ or *B*-”, then word *W* has right link *A* or left link *B*.
- {} - optionality. If “*W*: *A*+ & {*B*+}”, then after word *W* made right link *A*, it can have or not have link *B*.
- @ - unboundedness

Link



Result of parsing the sentence “*I am sitting on a chair*”

Transition-Based Dependency Parsing

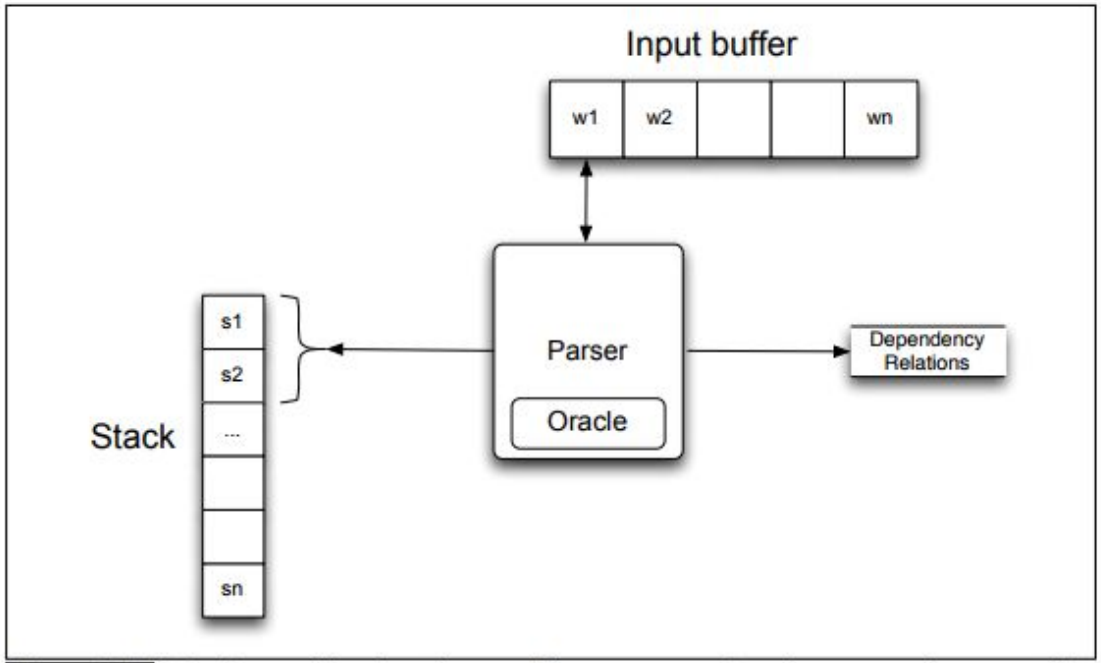


Figure 14.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

Transition-Based Dependency Parsing

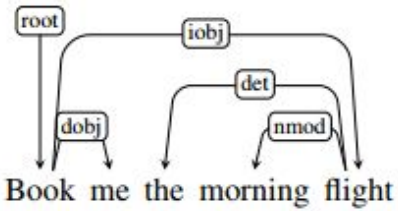
Actions:

- Assign the current word as the head of some previously seen word;
- Assign some previously seen word as the head of the current word;
- Or postpone doing anything with the current word, adding it to a store for later processing.

Transition operators:

- **LEFTARC**: Assert a head-dependent relation between the word at the top of stack and the word directly beneath it; remove the lower word from the stack;
- **RIGHTARC**: Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack;
- **SHIFT**: Remove the word from the front of the input buffer and push it onto the stack.

Transition-Based Dependency Parsing



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Figure 14.7 Trace of a transition-based parse.

Graph-based Dependency Parsing

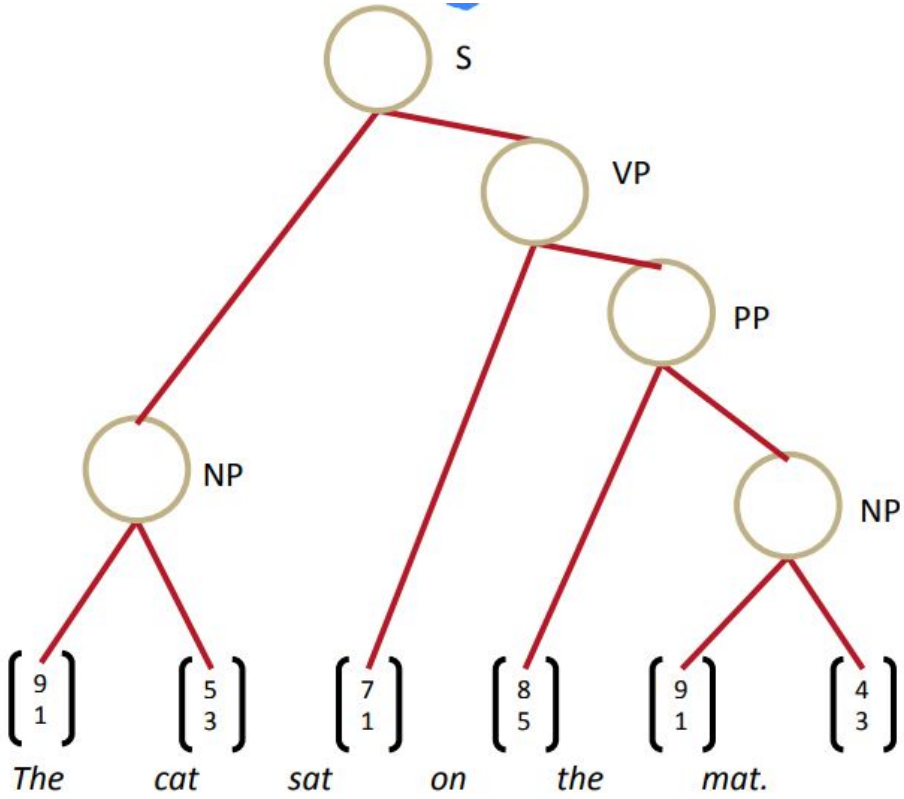
Graph-based approaches to dependency parsing search through the space of possible trees for a given sentence for a tree (or trees) that maximize some score.

More formally, given a sentence *S* we're looking for the best dependency tree in G_S , the space of all possible trees for that sentence, that maximizes some score.

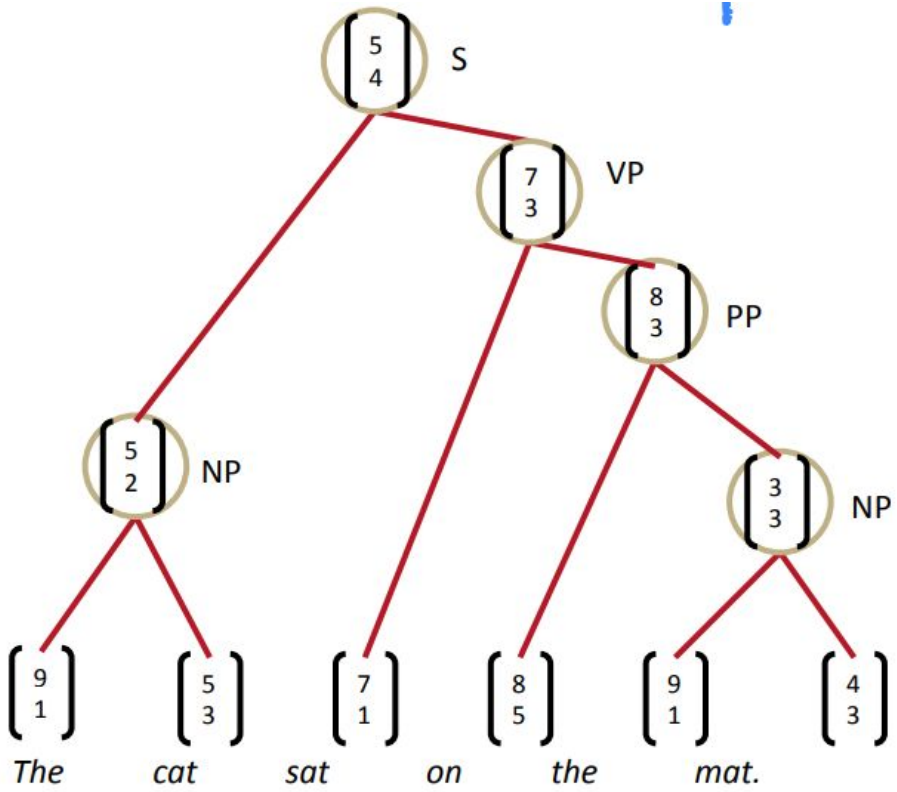
$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}}\, score(t, S)$$

$$score(t, S) = \sum_{e \in t} score(e)$$

Recursive Neural Networks

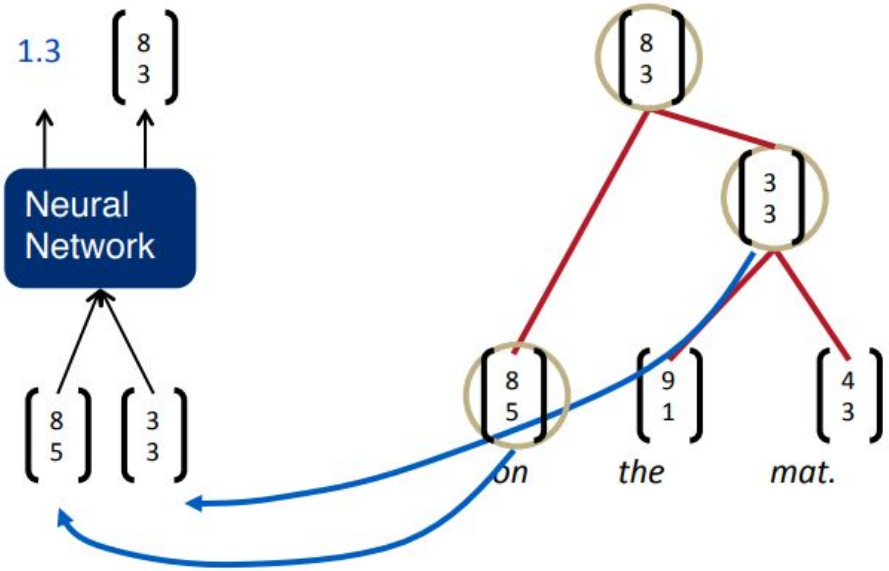


Recursive Neural Networks

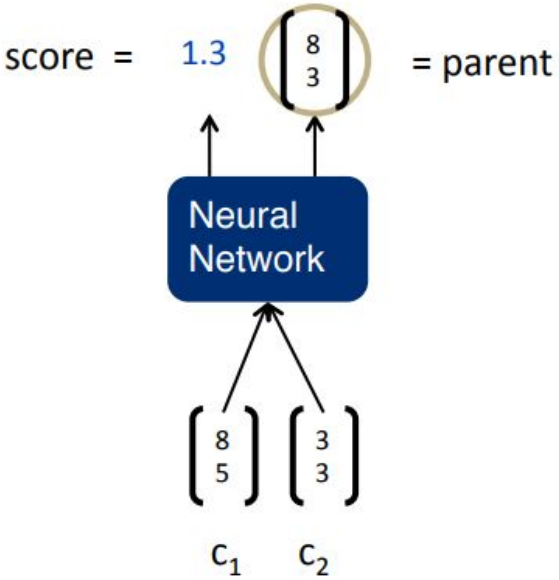


Recursive Neural Networks

- Inputs: two candidate children's representations
- Outputs:
  - 1. The semantic representation if the two nodes are merged.
  - 2. Score of how plausible the new node would be.



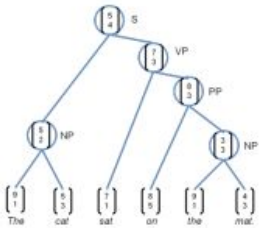
Recursive Neural Networks



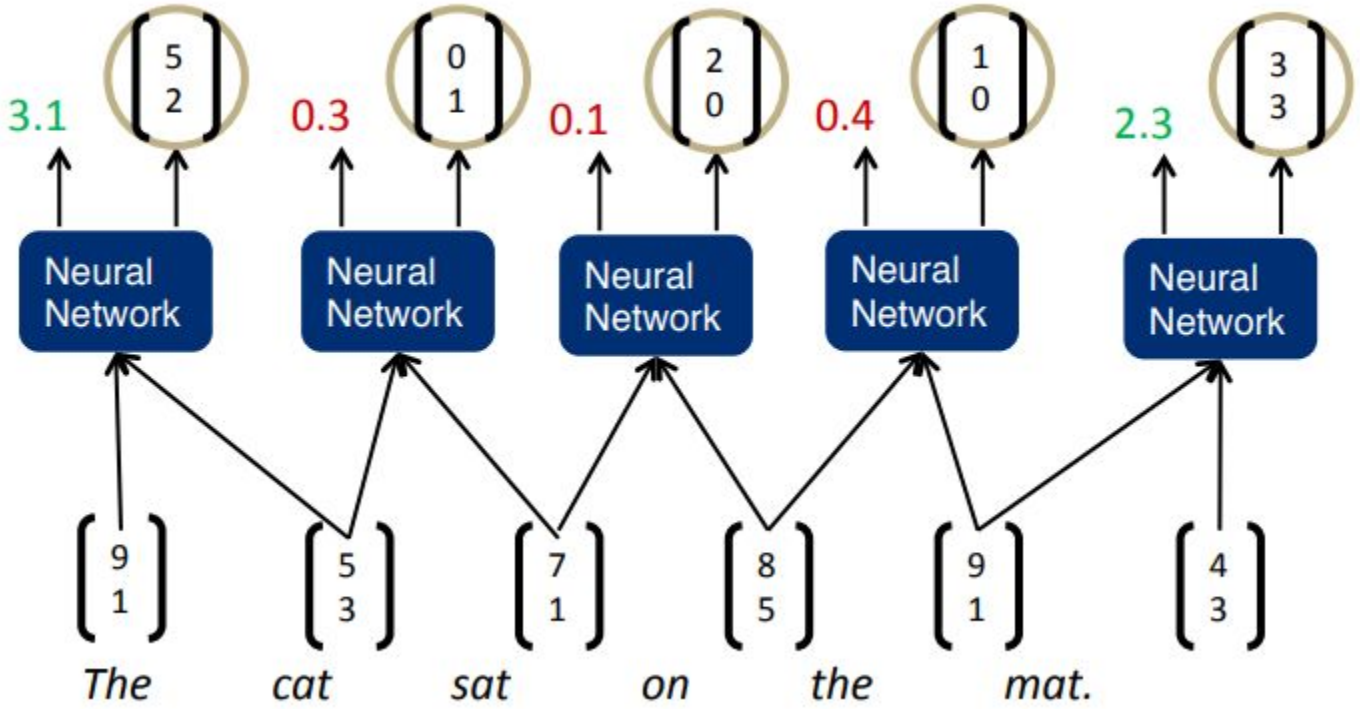
score = $U^T p$

$$p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right),$$

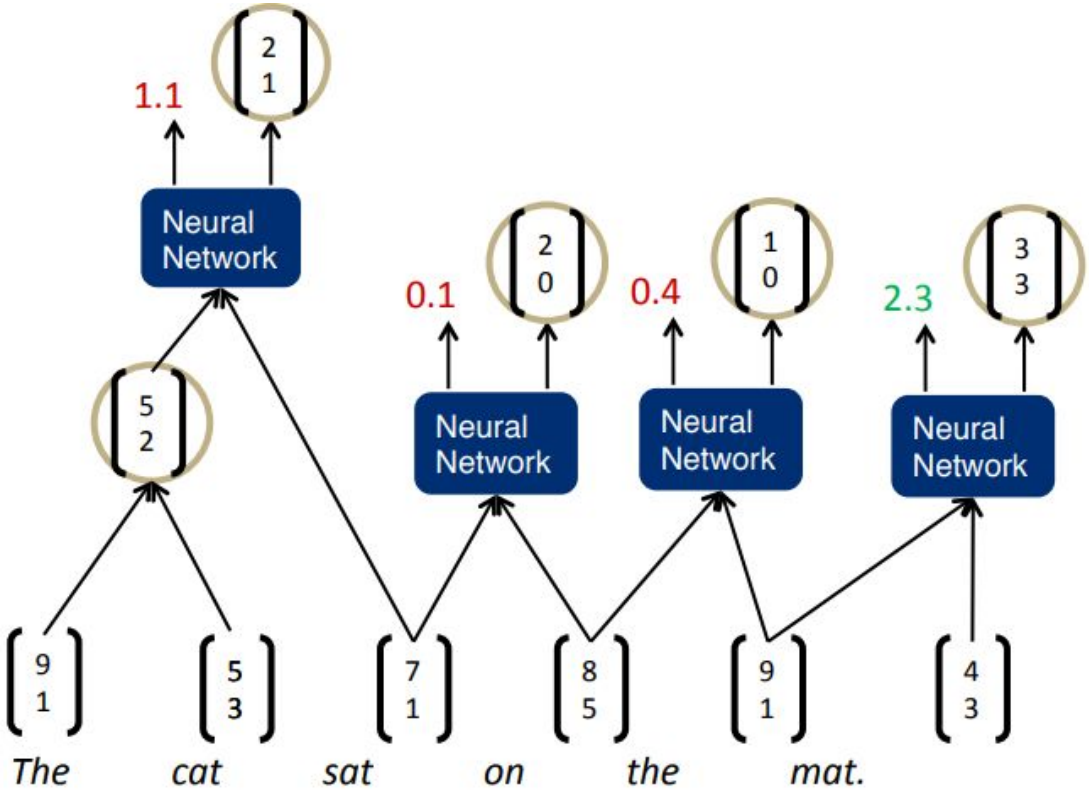
Same W parameters at all nodes of the tree



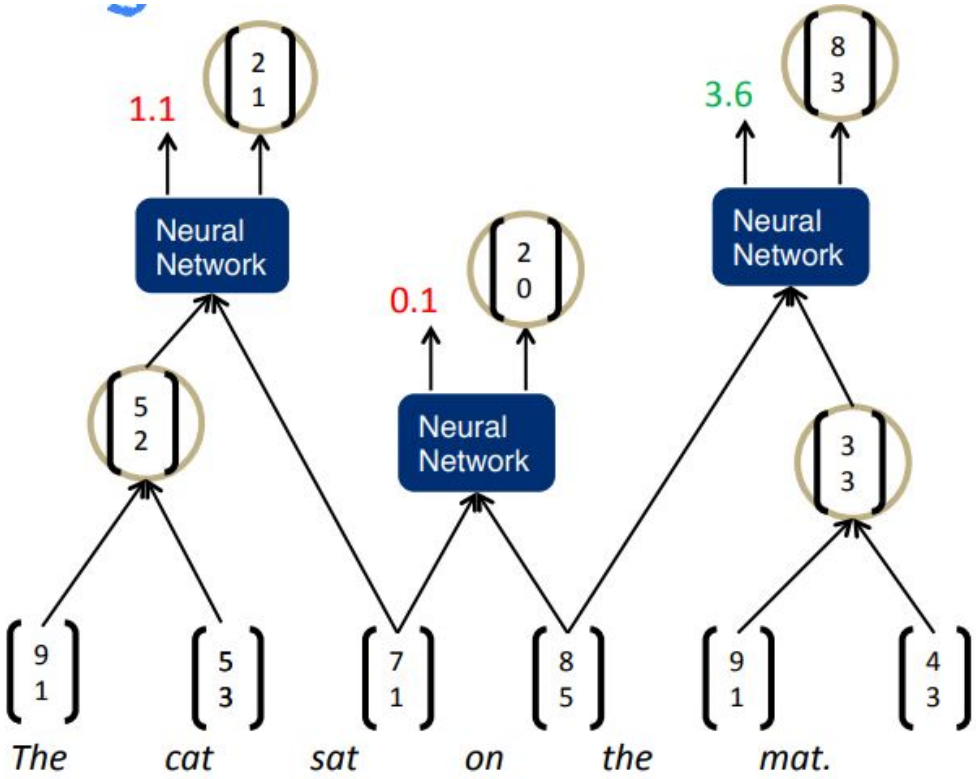
Recursive Neural Networks



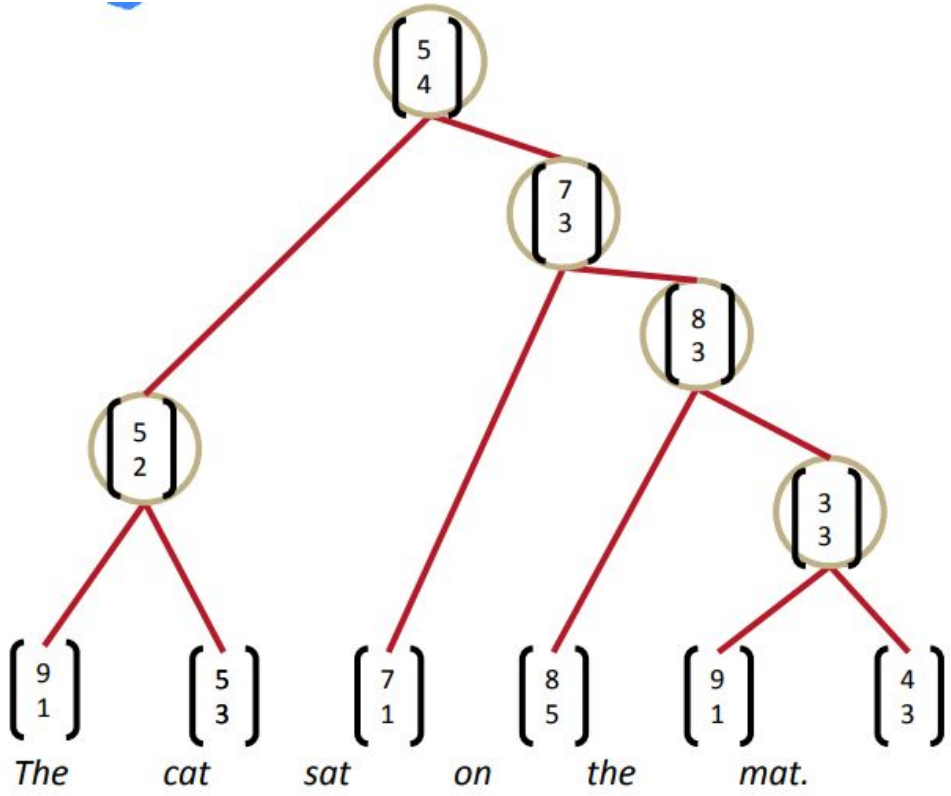
Recursive Neural Networks



Recursive Neural Networks



Recursive Neural Networks



Recursive Neural Networks

We can use it for dependency parsing as well!

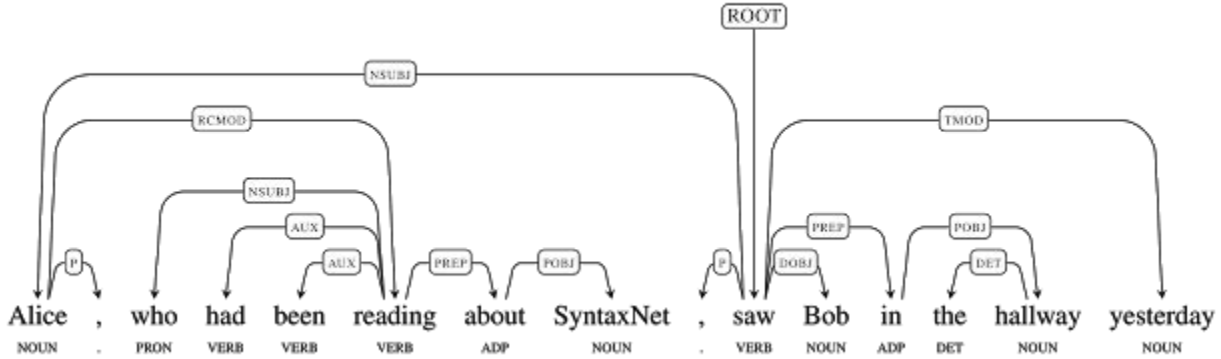
Try out some other features!

More: R.Socher et al. Deep Learning for NLP (without magic): https://nlp.stanford.edu/courses/NAACL2013/NAACL2013-Socher-Manning-DeepLearning.pdf

SyntaxNet

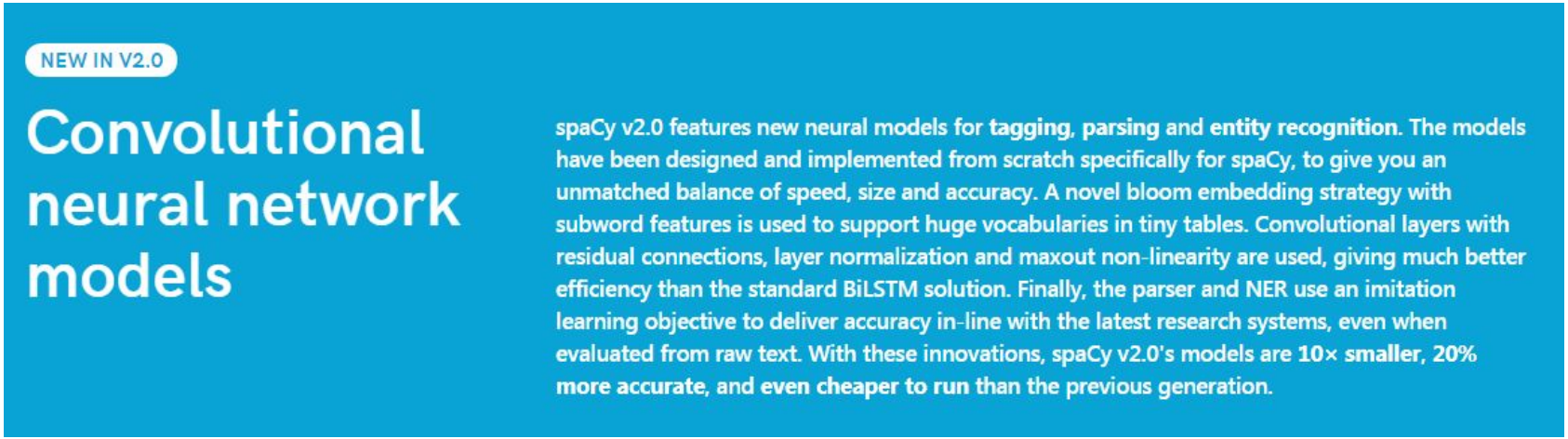
https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Given a sentence as input, it tags each word with a part-of-speech (POS) tag that describes the word's syntactic function, and it determines the syntactic relationships between words in the sentence, represented in the dependency parse tree.



Spacy

https://spacy.io/



Spacy

SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	**92.6**	*n/a* ⓘ
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

Spacy

SYSTEM	YEAR	TYPE	ACCURACY
spaCy v2.0.0	2017	neural	94.48
spaCy v1.1.0	2016	linear	92.80
Dozat and Manning	2017	neural	**95.75**
Andor et al.	2016	neural	94.44
SyntaxNet Parsey McParseface	2016	neural	94.15
Weiss et al.	2015	neural	93.91
Zhang and McDonald	2014	linear	93.32
Martins et al.	2013	linear	93.10

Spacy

	ABSOLUTE (MS PER DOC)			RELATIVE (TO SPACY)		
SYSTEM	TOKENIZE	TAG	PARSE	TOKENIZE	TAG	PARSE
spaCy	**0.2ms**	**1ms**	**19ms**	1x	1x	1x
CoreNLP	2ms	10ms	49ms	10x	10x	2.6x
ZPar	1ms	8ms	850ms	5x	8x	44.7x
NLTK	4ms	443ms	*n/a*	20x	443x	*n/a*

Evaluation

- **Exact match** (EM) — how many sentences are parsed correctly;
- **Labeled attachment score** (LAS) - the proper assignment of a word to its head along with the correct dependency relation;
- **Unlabeled attachment score** (UAS) - correctness of the assigned head, ignoring the dependency relation;
- **Label accuracy score** (LS) - the percentage of tokens with correct labels, ignoring where the relations are coming from;
- How well a system is performing on particular kind of dependency relation, for example *NSUBJ*, across a development corpus - use precision and recall !

Evaluation

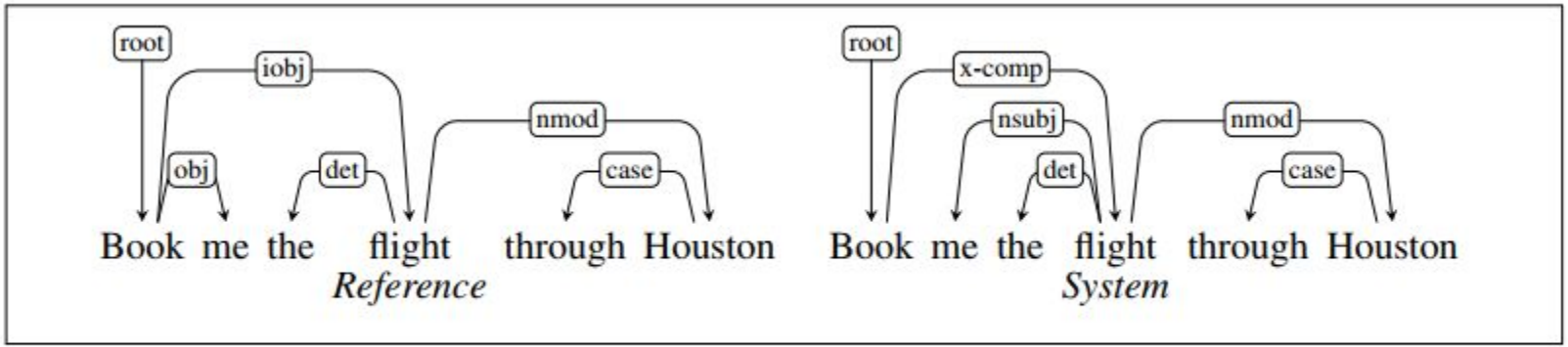
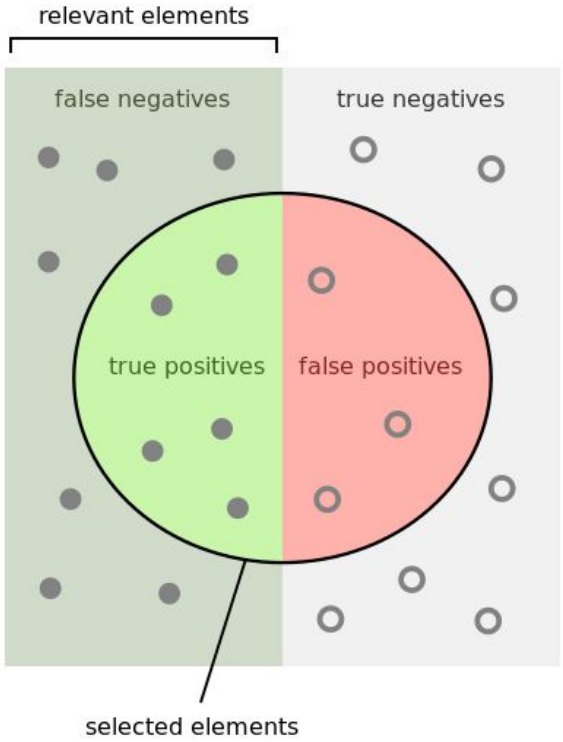


Figure 14.15 Reference and system parses for *Book me the flight through Houston*, resulting in an LAS of 3/6 and an UAS of 4/6.

Precision, Recall, F-measure



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

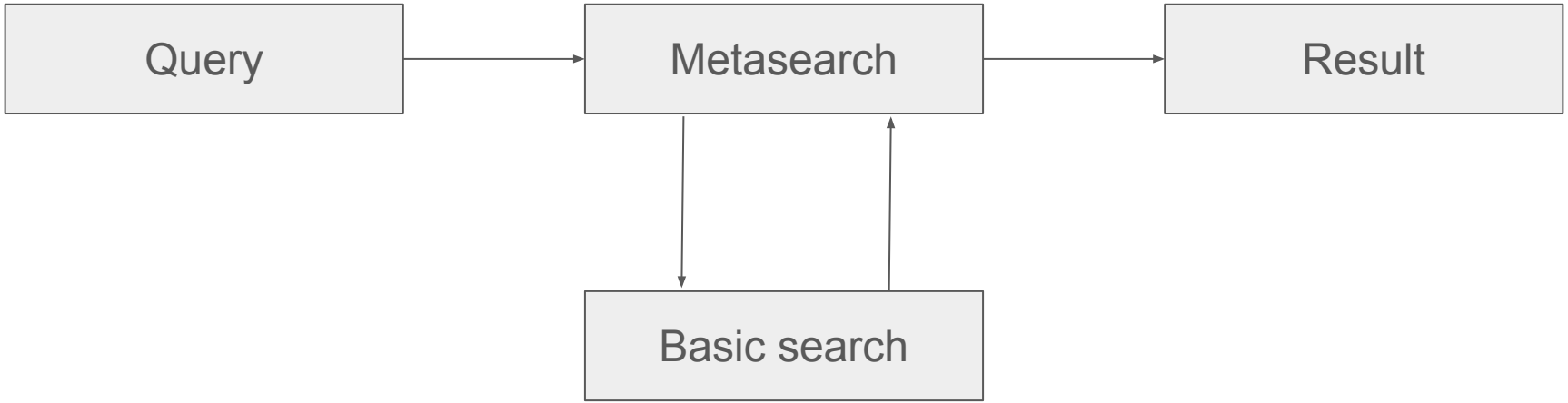
$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Comparison of russian syntax parsers (2012)

Parser	P	R	F1
Compreno	0.952	0.983	0.967
ETAP-3	0.933	0.981	0.956
SyntAutom	0.895	0.980	0.935
SemSyn	0.889	0.947	0.917
Dictum	0.863	0.980	0.917
Semantic Analyzer Group	0.856	0.860	0.858
AotSoft	0.789	0.975	0.872

MA and SA in search engines

Search engine is an ordered set of documents, designed for information storing and searching - texts (documents) or data (facts).



MA and SA in search engines

Index is a database containing information about words and their positions in documents which is in the Internet.

Indexing:

1. getting documents from web by links, their initial processing (MA and SA), buildup and saving the tables of key words frequencies etc.;
2. building linking metrics (e.g. PageRank);
3. building direct index;
4. adding new information to the index;
5. sorting by the metrics and building inverted index.

MA and SA in search engines

Basic search is a program, searching through the part of index.

Metasearch is a program, providing:

- receiving and processing of queries;
- selection of basic searches and sending them queries;
- caching pages with search results;
- documents ranking.

MA and SA in search engines

Order for query processing:

- estimation of current load of a search engine;
- analysis of query with metasearch;
- cache checking for the presence of a response;
- query processing with basic searches;
- response preparation by metasearch;
- building SERP (Search engine results page).

MA and SA in search engines

Ranking is a process of sorting search results when the documents, which are the most relevant to a query, are located at the top.

Relevant document is a document, whose the main subject corresponds to the semantic content of a query.

Thematic index of citing shows the authority of source with respect to others semantically close sources.

PageRank is the analogue of thematic index of citing in Google.

MA and SA in search engines

PageRank:

$$x_i = \alpha \sum_{j=1}^{N} \frac{x_j}{L(j)} + \frac{1-\alpha}{N}$$

L(j) - the number of outgoing links on page *j*;
x_j - PageRank of page *j*, linking on page *i*;
N - the total number of links on page *i*;
α - attenuation coefficient (usually 0.85)

MA and SA in search engines

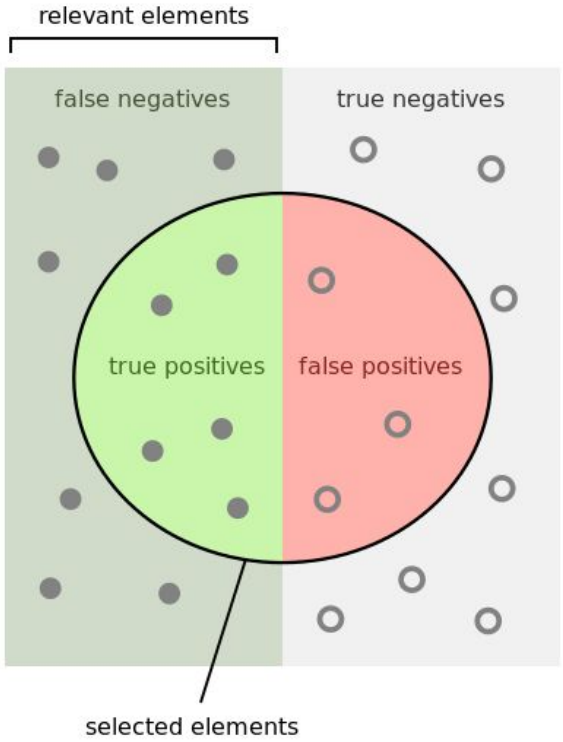
Morphological analyzer is used for indexing of all words in source text and of all words in user's query.

Syntactical analyzer is used for the internal query transformation and for further documents selection.

Operation of morphological analyzer is mapping $W \rightarrow L$, where W - set of all terms, L - set of all lemmas, and $|L| < |W|$. Thus the developers try to:

1. to increase recall;
2. to improve precision.

MA and SA in search engines



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

MA and SA in search engines

Query: What does the hypertension lead to*?*

Document 1*: Hypertension leads to disruption of blood supply in the tissues.*

Document 2: *Chronic lack of sleep leads to the hypertension.*

	Result	**Cause**
Query	*what*	*the hypertension*
Document 1	*disruption*	*the hypertension*
Document 2	*the hypertension*	*lack of sleep*

Syntax analysis can help!

Thank you for your attention!