Introduction to Parsing

Miguel Ballesteros

Algorithms for NLP Course.
7-11

Using some materials of Joakim Nivre from Uppsala University

Outline

- Input.
- Output.
- Mapping.
- Models.
- Evaluation

In 711

- Phrase-structure parsing:
  - CKY algorithm.
  - Earley's algorithm.
  - Shift-Reduce
  - PCFGs
  - Weighted CKY

In 711

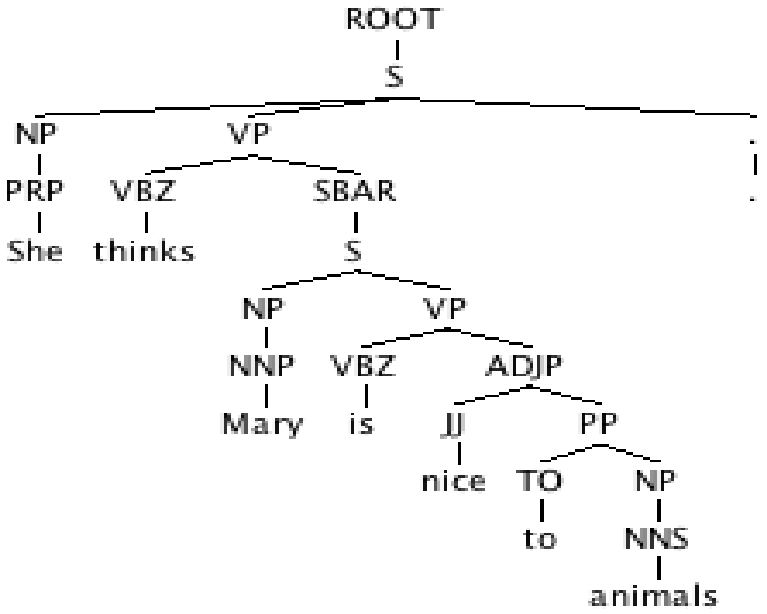
- Dependency parsing:
  - Intro to dependency linguistics.
  - Transition-based dependency parsing.
  - Graph-based dependency parsing.

Introduction

- Syntactic parsing is normally conceived as a structural prediction problem.
  - We map from an input X of sentences to an output space Y of syntactic representations.

Input

She thinks Mary is nice to animals →



Why Syntax Parsing?

- Parsing a sentence gives you
  - “who did what to whom?” in a sentence.
  - Linguistic structure in terms of Syntax.

  This is useful for:
  - Machine translation (syntax is essential for translation)
  - Web search (Google, for example)
  - Information extraction.
  - Many other tasks:
    - Text simplification.
    - Summarization
    - Etc.

Why is parsing hard?
Ambiguity

- Ambiguity. As in POS tagging or NER.

- Receives/inherits ambiguity from POS tagging.

Why is parsing hard?

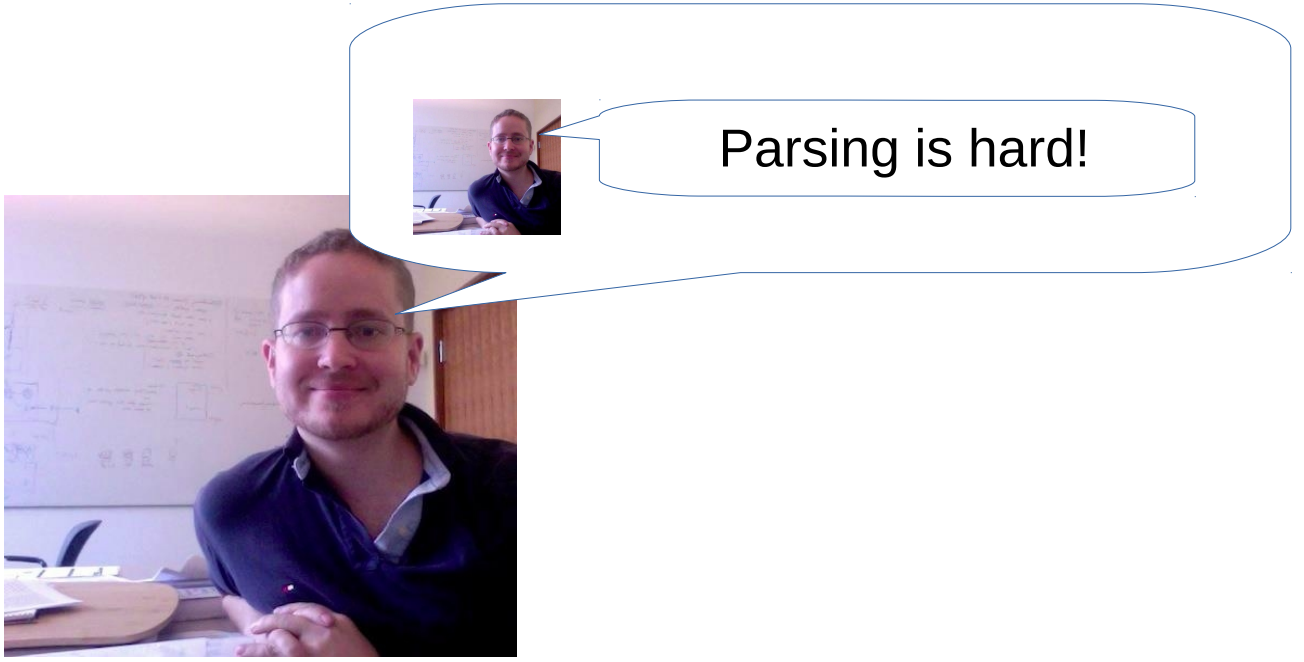
- Chris told us in the second lecture of the course that there is an infinite number of sentences in a natural language.

Why is parsing hard?

- Chris told us in the second lecture that there is an infinite number of sentences in a language.
- Remember the example of adding n- times “Chomsky said that” to any sentence.
- Chris said that parsing is hard!

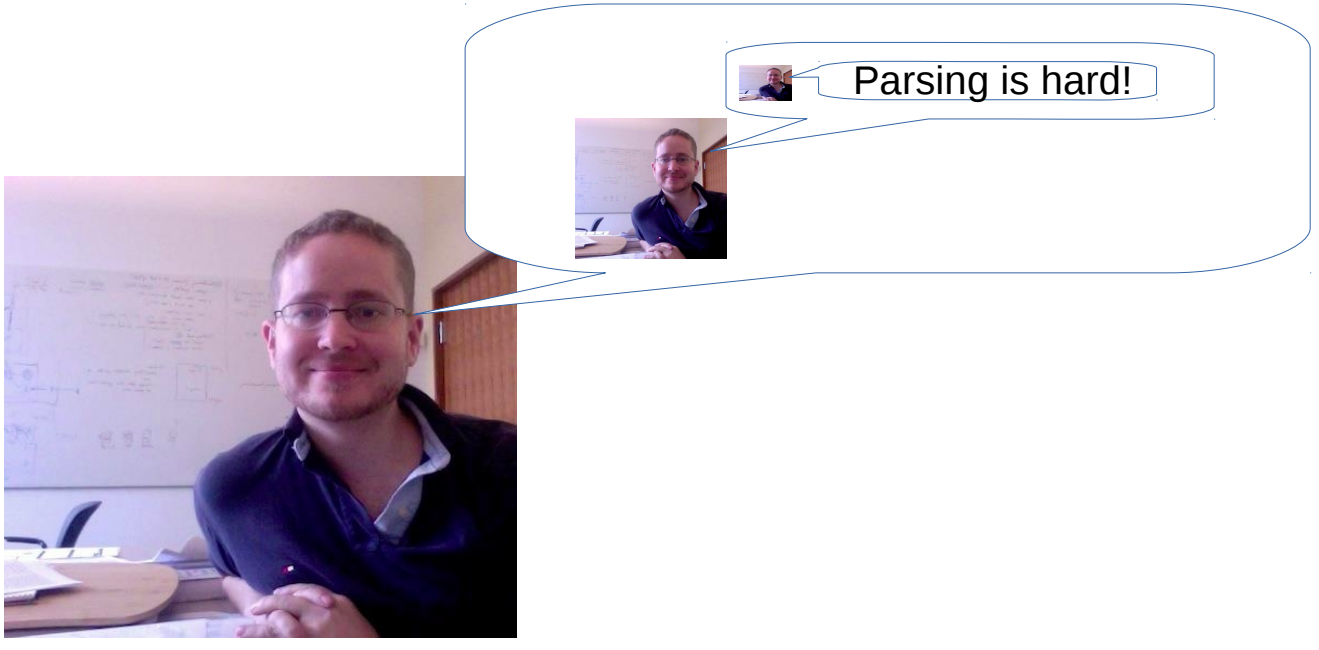
Why is parsing hard?

- Chris said that parsing is hard!
- Chris said that Chris said that parsing is hard!



Why is parsing hard?

- Chris said that parsing is hard!
- Chris said that Chris said that parsing is hard!
- Chris said that Chris said Chris said that parsing is hard!



Why is parsing hard?

- We have to parse sentences that we have never seen when we trained our parser.

  Or even harder:

- **We have to parse sentences that have never been written before.**

Why is Parsing hard?

- The man saw the girl with a **telescope**.
  - so... who had the telescope?

Why is parsing hard? Ambiguity

- The man saw the girl with a **telescope**.
  - so... who had the telescope?

Input

- We generally assume that an input $x \in \mathbf{X}$ is a sentence consisting of a sequence $x_1 \ldots x_n$ of tokens.

- How do we delimit sentences?
- How do we split sentences into tokens?
- What properties do tokens have?

Input

- In carefully edited “western” script, the delimitation of sentences is usually straightforward.
- In other writing systems, and in certain text genres like email and twitter, this task can be much more challenging.

In spoken language, many researchers would question that the sentence is a relevant unit at all.

Input

- In syntactic parsing research, it is generally assumed that the delimitation of sentences is not part of the parsing task itself.
- However, parsing presupposes adequate sentence segmentation.

- Parsing results will be much worse if sentence segmentation is not correct.

Input

Dr. Frederking sent an application last week.

- If we use a simple sentence splitting that assumes that dots divide sentences, then we have a problem...
  - Dr . (1st sentence).
  - Frederking sent an application last week. (2nd sentence).
- And we just want one sentence!
  - Dr. Frederking sent an application last week.

Input

- Whereas alphabetic scripts often contain reliable clues to word boundaries.
- The word segmentation problem in written Chinese is much more challenging.

Input

- In some morphologically rich languages, word forms can be quite complex and include elements that in other languages would be realized as independent word forms.

Input

- Whereas alphabetic scripts often contain reliable clues to word boundaries.
- The word segmentation problem in written Chinese is much more challenging.
- In some morphologically rich languages, word forms can be quite complex and include elements that in other languages would be realized as independent word forms.

- **Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsinizcesine**

Input

ev – (the) house

evler – (the) houses

evleriniz – your houses

evlerinizden – from your houses

evlerinizdendi – (he/she/it) was from your houses

- Turkish is an agglutinative language.
- Turkish 2006 treebank and Turkish 2007 treebank.

Input

- The question of whether word forms should be subjected to morphological segmentation prior to parsing is related to the more general question of what linguistic analysis of tokens (if any) should be carried out before parsing begins.

Input

- there are systems (most) that presuppose that the tokens have been annotated for parts of speech, lemma, and morphosyntactic properties like case, number, tense, aspect

They	**told**	**him**	**a**	**story**	**.**
pos: prp	pos: vb	pos: prp	pos:dt	pos: nn	pos: p
numb: plu	–	numb: sing	–	numb: sing	–
–	tense=past	–	–	–	–
pers: 3rd	–	–	–	–	–

Input

- we find systems that take only the raw tokens (as sequences of characters) as input to parsing and which may or may not provide the additional annotation as output of the parsing process.

Input

- we find systems that take only the raw tokens (as sequences of characters) as input to parsing and which may or may not provide the additional annotation as output of the parsing process.

In any case, it is important to keep these differences in mind when comparing different parsing systems.

Output

- An input sentence $x \in X$ should be mapped to a syntactic representation $y \in Y$, but what counts as a syntactic representation of a sentence?

Output

- An input sentence $x \in X$ should be mapped to a syntactic representation $y \in Y$, but what counts as a syntactic representation of a sentence?

- Different linguistic theories have provided different answers to this question.
  - in terms of what properties and relations to represent (for example, **dependency or constituency**).
  - Or in how to delimit syntax from morphology.
  - Or in how to delimit syntax from semantics.

Output - Constituency

- Historically, the most common type of representation used in parsing is based on the notion of **constituency**.

Output - Constituency

- Historically, the most common type of representation used in parsing is based on the notion of **constituency**.

- In a constituent structure (or phrase structure), a sentence is recursively decomposed into smaller segments that are categorized according to their internal structure into noun phrases, verb phrases, etc.

Output - Constituency

- Historically, the most common type of representation used in parsing is based on the notion of **constituency**.

- In a constituent structure (or phrase structure), a sentence is recursively decomposed into smaller segments that are categorized according to their internal structure into noun phrases, verb phrases, etc.
- **CFGs!**

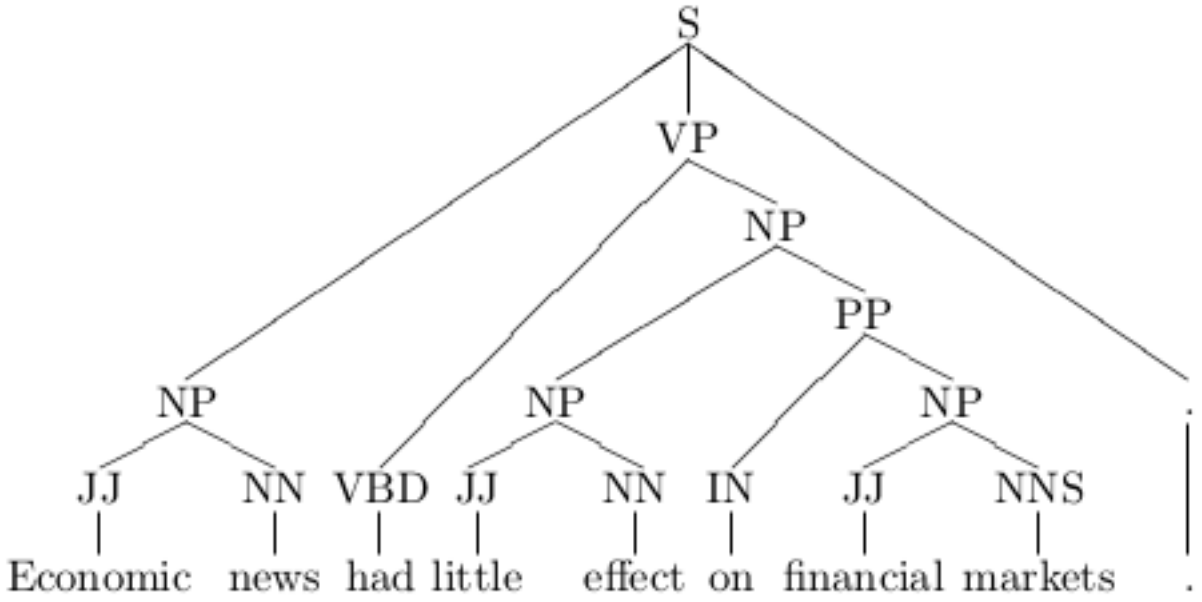
Output - Constituency

- **Constituent** structures are naturally induced by CFGs (Chomsky, 1956)
- Many theoretical frameworks:
  - Lexical Functional Grammar (**LFG**) (Kaplan & Bresnan, 1982; Bresnan,2000),
  - Tree Adjoining Grammar (**TAG**) (Joshi, 1985, 1997)
  - Head-Driven Phrase Structure Grammar (**HPSG**) (Pollard & Sag, 1987, 1994).

Output - Constituency

- They are also widely used in annotation schemes for treebanks, such as
  - The Penn Treebank scheme for English (Marcus et al., 1993, 1994),
  - Adaptations of this scheme that have been developed for Chinese (Xue et al., 2004), Korean (Han et al., 2002), Arabic (Maamouri & Bies, 2004), and Spanish (Moreno et al., 2003).
  - And other languages, see SPMRL 2013 data sets, for example.

Output - Constituency



Output - Dependency

- Another kind of representation that has gained popularity in recent years is instead based on the notion of **dependency**.

Output - Dependency

- Another kind of representation that has gained popularity in recent years is instead based on the notion of **dependency**.

- In a dependency structure, a sentence is analyzed by connecting its words by binary asymmetrical dependency relations, and words are categorized according to their functional role into *subject, object, etc*.

Output - Dependency

- **Dependency** structures are adopted in theoretical frameworks such as
- Functional Generative Description (Sgall et al., 1986)
- Meaning-Text Theory (Mel’ˇcuk, 1988)

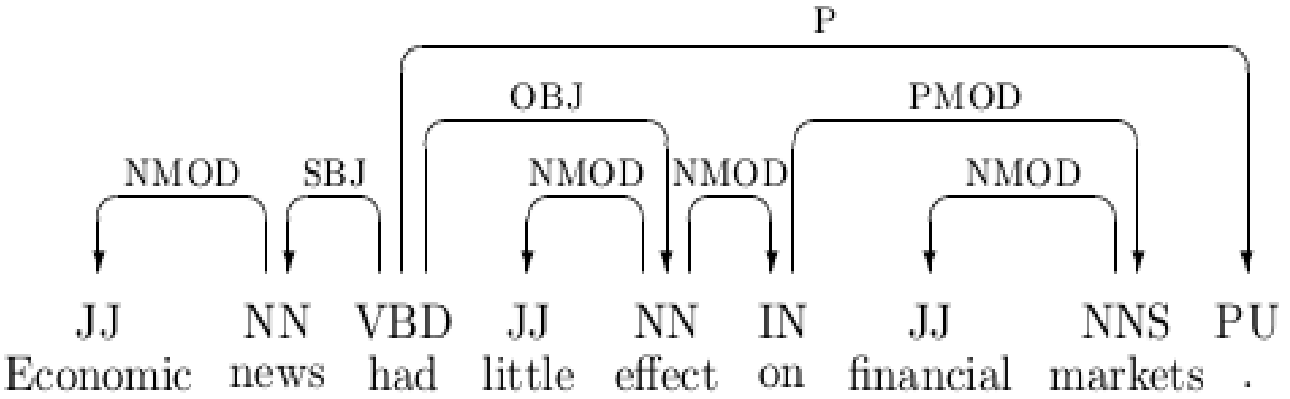
They are used for treebank annotation especially for languages with free or flexible word order.

Output - Dependency

- Prague Dependency Treebank of Czech (Hajiˇc et al., 2001; Bˇohmov á et al., 2003),
- Arabic (Hajiˇc et al., 2004), Basque (Aduriz et al., 2003), Danish (Kromann, 2003), Greek (Prokopidis et al., 2005), Russian (Boguslavsky et al., 2000), Slovene (Dˇzeroski et al., 2006), Turkish (Oflazer et al., 2003), and other languages.
- Dependency version of the Penn treebank.

Etc.

Output - Dependency



Output – Syntactic (+Semantic)

- A third kind of syntactic representation is found in categorial grammar, which connects syntactic (and semantic) analysis to inference in a logical calculus.

- In statistical parsing, categorial grammar is mainly represented by Combinatory Categorial Grammar (CCG) (Steedman, 2000),
- CCGbank (Hockenmaier & Steedman, 2007), a reannotation of the Wall Street Journal section of the Penn Treebank.

Mapping

- The parsing problem.
- how to characterize the mapping **X** → **Y** ?
  - Being X the input and Y the output.
  - Is it a function or a relation?
  - What relation should hold between a sentence and its syntactic representation(s)?

Mapping (grammaticality - 1)

- Historically, the notion of parsing was intimately tied to the notion of *grammaticality*,
- Task of the parser is twofold:
  - to decide whether a sequence of tokens was a **grammatical** sentence
  - to derive every valid syntactic representation for the sentence

Mapping (grammaticality - 2)

- This implies that the mapping is a **relation**, not a function, where:
  - ambiguous sentences are mapped to more than one representation
  - ungrammatical sentences (token sequences) are mapped to nothing.
- The task of ***disambiguation***, that is, of selecting the contextually most appropriate representation in cases of ambiguity, is on this view not part of parsing itself.

Mapping (grammaticality - 3)

- Important distinction:

  1- input x being grammatical or not with respect to a specific formal grammar G (say, a **CFG**)

  2- and x being grammatical in a given natural language (say, English or Spanish).

- 1 is a very well defined problem.
- 2 is not. Since there is no **“exact” grammar** of English or Spanish.

And this is why parsing is hard.

We hope for the best :-) : solving 1 is a good approximation of solving 2.

Mapping (optimality - 1)

- More recent.
- The task of a parser is then to return the contextually most appropriate representation for an input sentence,
- There could be many other “grammatical” or close to grammatical representations.
- A parser is supposed to return the output y* that maximizes some mathematical function $\boldsymbol{f_M : X \times Y \rightarrow \Re^2}$

$$y^* = \underset{y}{\operatorname{argmax}}\, f_M(x, y)$$

Mapping (optimality - 2)

- Difficult problem because we lack knowledge of the true optimization function used by native speakers of the language.

- It is a basic assumption in most of current parsing research that we can approach it with annotated data in form of treebanks.

Mapping (optimality - 3)

- We assume that the task of a parser is to map a sentence x ϵ *X* to an optimal syntactic representation y ϵ *Y*,
  - And we expect that there is only one representation.

Model

- A syntactic parser is based on a mathematical model M relating the input space X to the output space Y.

- M must provide a ranking or scoring of possible outputs for a given input.

Model

- It is often natural to see a parsing model as
  M = (GEN, EVAL)
  - GEN: A generative component GEN that maps an input x to a set of candidate analyses
    $\{y_1, \ldots, y_k\}$, that is, GEN(x) $\subseteq$ Y (for x $\in$ X).
  - An evaluative component EVAL that ranks candidate analyses via a numerical scoring
    scheme, that is, EVAL(y) $\in \Re$ (real numbers)

    (for y $\in$ GEN(x)).

Model

- Thus, what we want to find is

$$y^* = \underset{y}{\operatorname{argmax}}\, f_M(x, y) = \underset{y \in GEN(x)}{\operatorname{argmax}}\, EVAL(X, Y)$$

- *B*oth the generative and the evaluative component may include parameters that need to be estimated from empirical data using statistical inference.

Model

- Given that we have constructed a parsing model (with or without statistical learning), we need an efficient way of constructing and ranking the candidate analyses for a given input sentence.

- This is the ***inference*** problem for a parsing model.

Evaluation

- The most common way of evaluating the *accuracy* of a parser is to run the parser on a sample of sentences:
  - $T = \{(x^1 , y^1), . . . , (x^m , y^m)\}$ :: the test set.

  Assuming that the treebank annotation y^i for each sentence x^i is the preferred analysis.

Evaluation

- **The test set should not be touched during the creation and optimization of the system.**

Evaluation

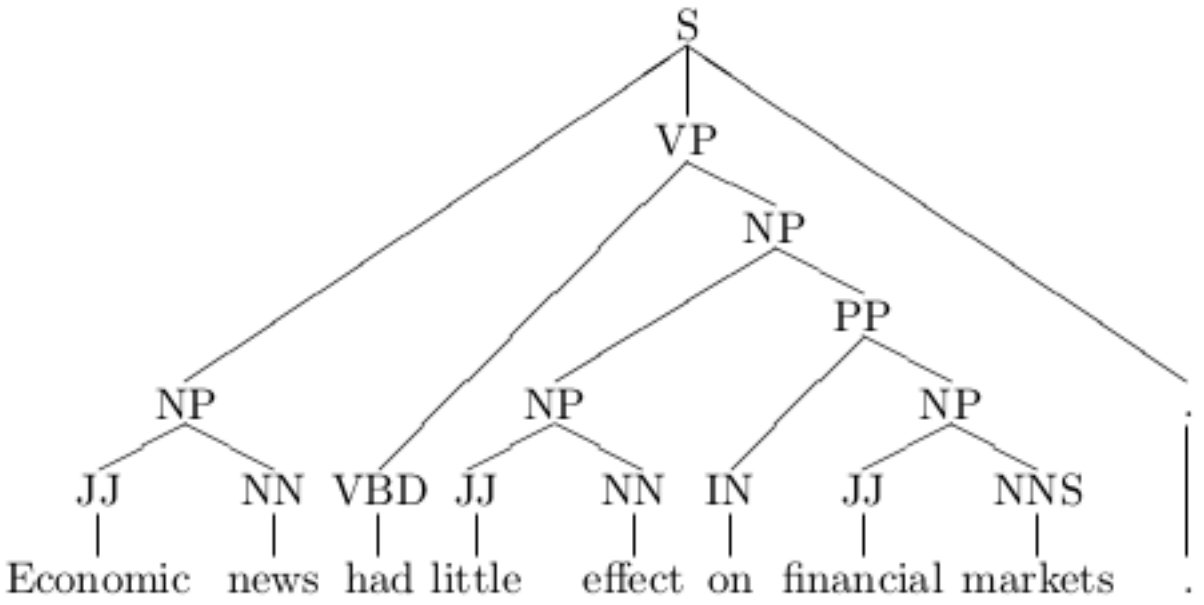
- The simplest way of measuring the test set accuracy is to use the Exact Match score:
  - Counts the exact number of sentences in which the parser output is identical to y^i

Evaluation

- The simplest way of measuring the test set accuracy is to use the Exact Match score:
  - Counts the exact number of sentences in which the parser output is identical to y^i
  - It is a crude metric.
  - Users might like it, though.

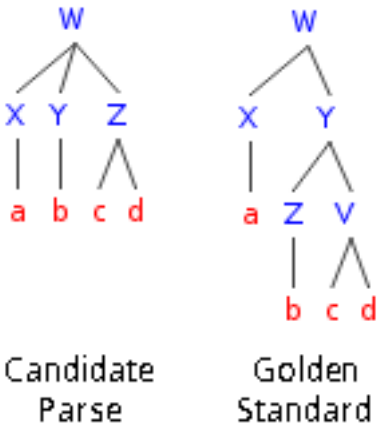
Evaluation

- For constituency parsers:
  - **PARSEVAL** metrics (Black et al., 1991; Grishman et al., 1992)
  - consider the number of matching constituents between the parser output and the gold standard.



Evaluation

Candidate	Gold
X: a	X:a
Y: b	Z: b
Z: cd	V: cd
--	Y: b c d
W: a b c d	W: a b c d

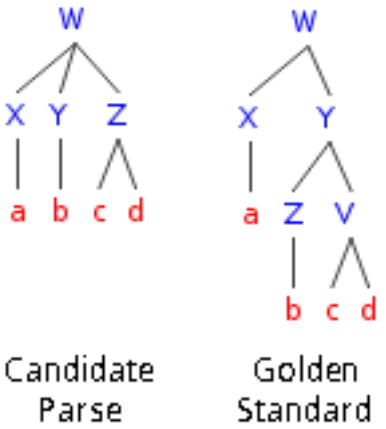


Candidate Parse

Golden Standard

Evaluation

Candidate	Gold
X: a	X:a
Y: b	Z: b
Z: cd	V: cd
--	Y: b c d
W: a b c d	W: a b c d

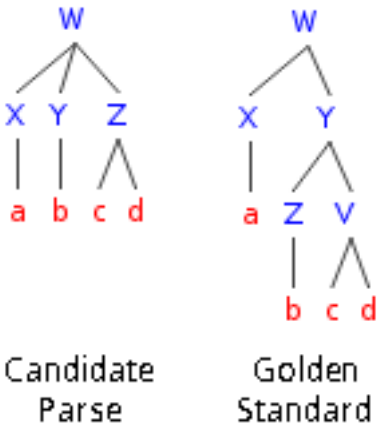


- P … we count la … the co …
- Tl …
  - For unlabeled, it would be 100% precision.

precision: number of correct constituents (yield) in parser output divided by number of constituents in the parser output

Evaluation

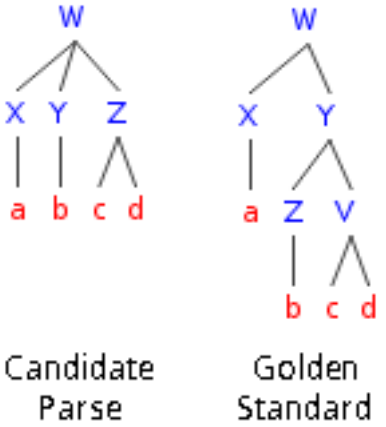
Candidate	Gold
X: a	X:a
Y: b	Z: b
Z: cd	V: cd
--	Y: b c d
W: a b c d	W: a b c d



- Precision: all have correct yield but we count labels. 2 have the correct label and the correct yield.
- This means 50% precision. 2/4
  - For unlabeled, it would be 100% precision.

Evaluation

Candidate	Gold
X: a	X:a
Y: b	Z: b
Z: cd	V: cd
-- !!	Y: b c d
W: a b c d	W: a b c d



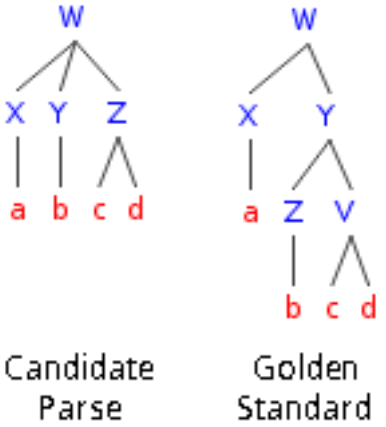
- R yields in g
- T

recall: number of constituents from the gold standard (yield) that can be found in the parser output divided by the number of constituents in the gold standard

– call.

Evaluation

Candidate	Gold
X: a	X:a
Y: b	Z: b
Z: cd	V: cd
-- !!	Y: b c d
W: a b c d	W: a b c d



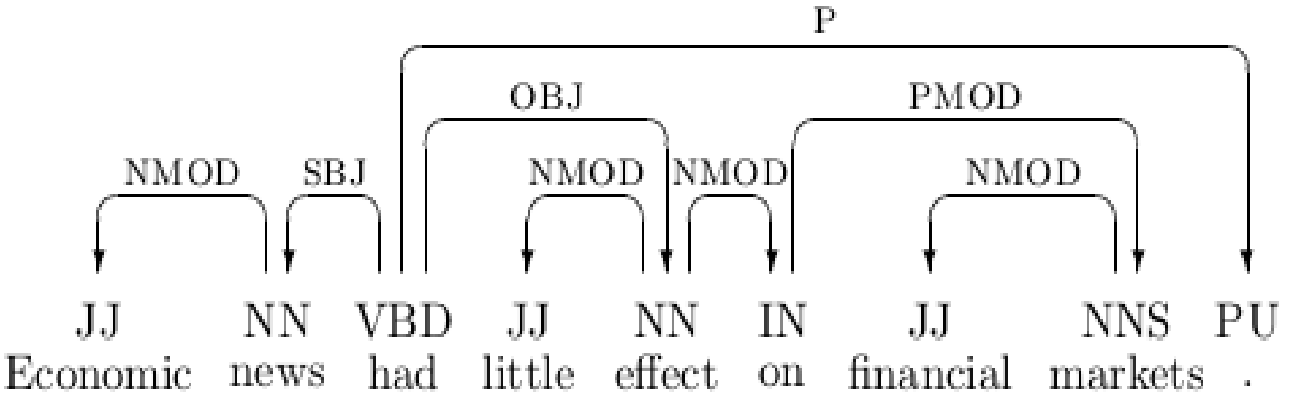
- Recall: there is a missing subtree. 5 yields in gold.
- This means 40% recall. 2/5
  - For unlabeled, it would be 80% recall.

Evaluation

- Complaints about Parseval:
  - Rewards shallow/safe analyses better than those that make more claims but a few mistakes.
  - Especially with corpora, punishes parsers that provide more information than necessary.
  - Some "single" errors can hurt the score repeatedly, for example a single misplaced node may trigger multiple crossing brackets and incorrect yields.
  - Weights all nodes evenly, rather than making crucial semantical relations more important.

Evaluation

- For **dependency** parsers:
  - Attachment score. (Buchholz & Marsi, 2006)
  - Measures the proportion of words in a sentence that are attached to the correct head.
  - (could be labeled)



Evaluation

	Head-Gold	Label-Gold	Head-System	Label-System
1 . Parsers	2	SBJ	2	DOBJ
2. are	0	ROOT	0	Root
3. cool	2	PRD	1	PRD
4. .	2	Punct	1	Punct

Evaluation

	Head-Gold	Label-Gold	Head-System	Label-System
1 . Parsers	2	SBJ	2	DOBJ
2. are	0	ROOT	0	Root
3. cool	2	PRD	1	DOBJ
4. .	2	Punct	1	Punct

- Unlabeled attachment score:
  - 2 heads out of 4 correct:
  - 50% UAS

Evaluation

	Head-Gold	Label-Gold	Head-System	Label-System
1 . Parsers	2	SBJ	2	DOBJ
2. are	0	ROOT	0	Root
3. cool	2	PRD	1	PRD
4. .	2	Punct	1	Punct

- Labeled attachment score:
  - 1 head and label out of 4 correct:
  - 25% LAS

Evaluation

	Head-Gold	Label-Gold	Head-System	Label-System
1 . Parsers	2	SBJ	2	DOBJ
2. are	0	ROOT	0	Root
3. cool	2	PRD	1	PRD
4. .	2	Punct	1	Punct

- Label accuracy:
  - 3 labels out of 4 correct:
  - 75% LA

Evaluation

- Time and space efficiency are also important.
- processing speed and memory consumption, can be critical and therefore often need to be evaluated empirically

- A faster parser with reasonable results could be (I'd say **IS**) more interesting than a very slow but accurate parser.

Parsing algorithms

- CKY (bottom-up)
- Earley (top-down)
- Shift-Reduce

- Transition-based dependency parsers.
- Graph-based dependency parsers.