Dependency / Link Parsing

Sebastian Dufner

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Chair for Algorithms and Data Structures

11.01.2012

Outline

Outline

Outline

Outline

Outline

Classification

Constituent, Dependency and Link Parsing are very similar

- Constituent Parsing: phrase structure
- Dependency Parsing: words are syntactic functions
- Link Parsing: relations between words

Classification

Constituent, Dependency and Link Parsing are very similar

- Constituent Parsing: phrase structure
- Dependency Parsing: words are syntactic functions
- Link Parsing: relations between words

Classification

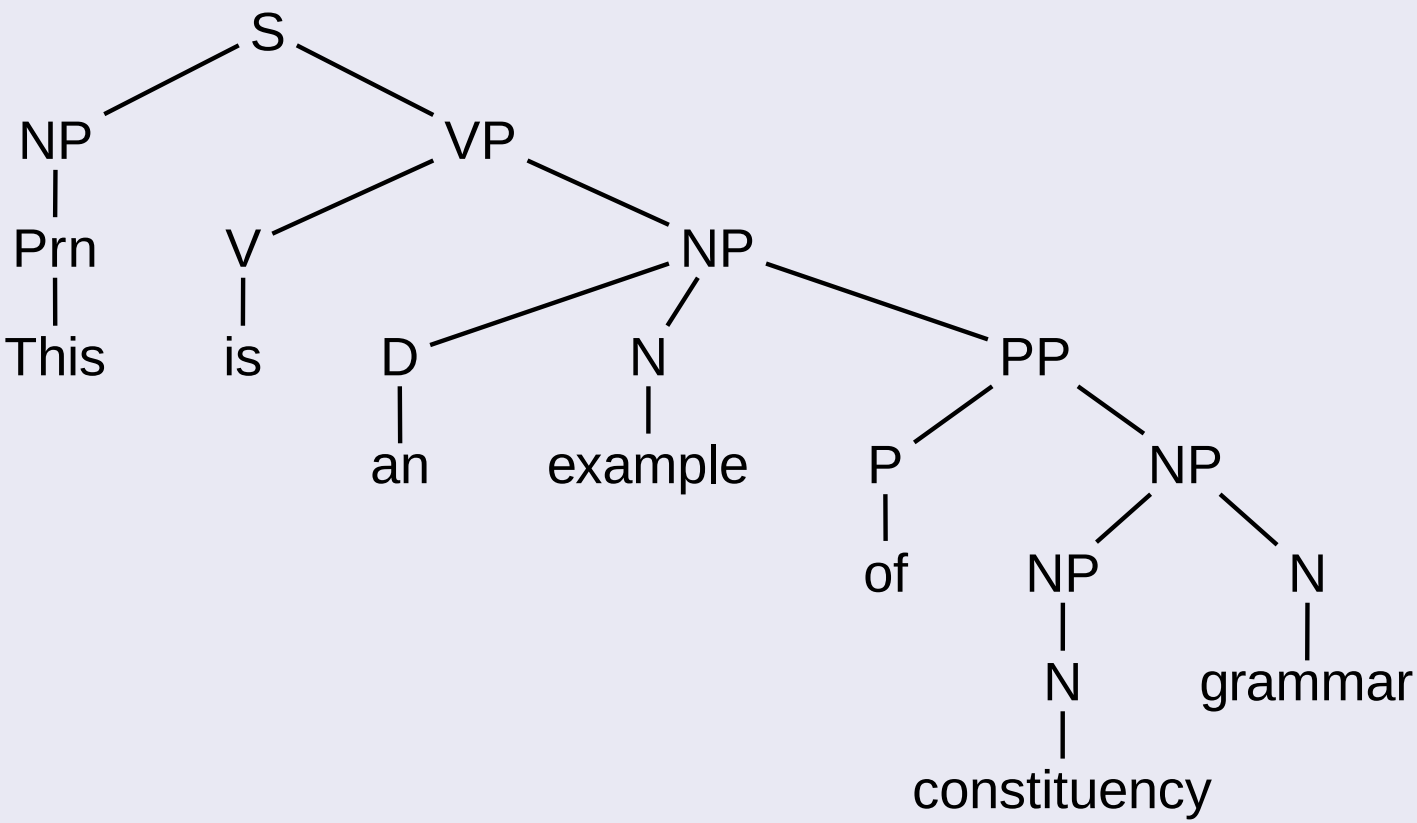
Constituent, Dependency and Link Parsing are very similar

- Constituent Parsing: phrase structure
- Dependency Parsing: words are syntactic functions
- Link Parsing: relations between words

Outline

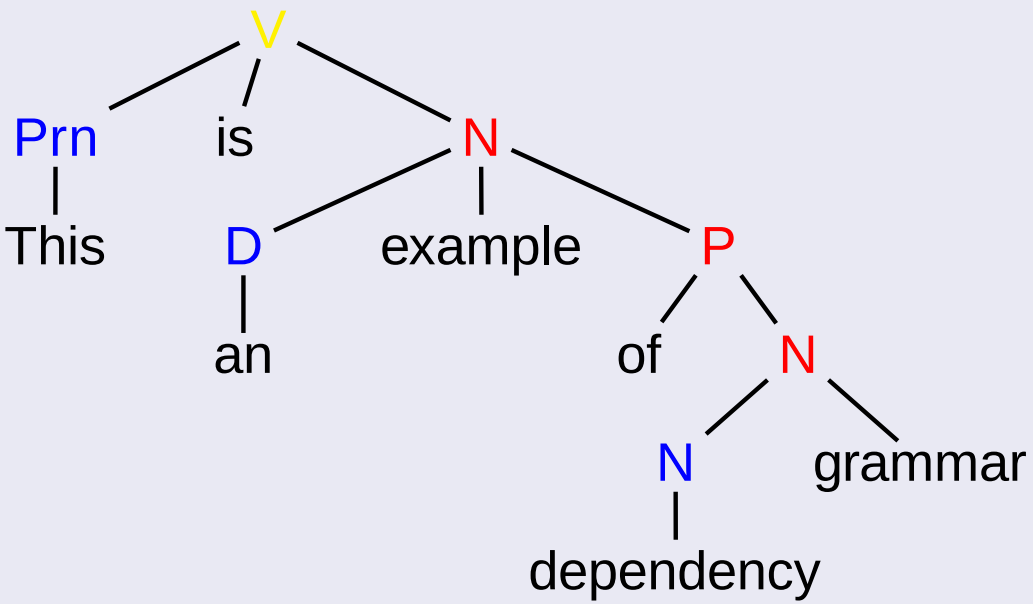
Examples

Constituency Parsing



Examples

Dependency Parsing



- yellow: independent
- blue: predependent
- red: postdependent

Outline

Rules for Dependency Trees

- Dependent needs head
- Head may need presence of dependent
- Tree (directional acyclic graph)
  $\longrightarrow$ main verb as root

A word and all its dependents form a sentence.

Rules for Dependency Trees

- Dependent needs head
- Head may need presence of dependent
- Tree (directional acyclic graph)
  $\longrightarrow$ main verb as root

A word and all its dependents form a sentence.

Rules for Dependency Trees

- Dependent needs head
- Head may need presence of dependent
- Tree (directional acyclic graph)
  $\longrightarrow$ main verb as root

A word and all its dependents form a sentence.

Outline

Basic Concept

Parser looks successively onto words

- Attaches them to tree as soon as a match is found
- Very similar to human behavior
- Popular in many languages

Basic Concept

Parser looks successively onto words

- Attaches them to tree as soon as a match is found
- Very similar to human behavior
- Popular in many languages

Basic Concept

Parser looks successively onto words

- Attaches them to tree as soon as a match is found
- Very similar to human behavior
- Popular in many languages

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Assumptions

- **Unity**: Result is a tree
- **Uniqueness**: Only one head per word
- **Adjacency**: If A depends on B, all words between A and B are subordinate to B
  $\longrightarrow$ equiv. to “no crossing branches” in constituency
- **Word-at-a-time**: Only one word at a time, attaching them when encountered
- **Left-right pass**: If not forced to backtrack, parser makes only one pass from left to right
- **Eagerness**: Parser attaches words as early as possible

Outline

Rules

Local Rules:

- Words are like blocks with connectors.
- Connectors must be connected with conntectors of the same type.
- Connectors point either to left or right.

Rules

Local Rules:

- Words are like blocks with connectors.
- Connectors must be connected with conntectors of the same type.
- Connectors point either to left or right.

Rules

Local Rules:

- Words are like blocks with connectors.
- Connectors must be connected with conntectors of the same type.
- Connectors point either to left or right.

Rules

Global Rules:

- Connectors must not cross.
- All words in a sentence have to be attached to the tree.

Valid sentence: All local and global rules can be applied.

Rules

Global Rules:

- Connectors must not cross.
- All words in a sentence have to be attached to the tree.

Valid sentence: All local and global rules can be applied.

Rules

Global Rules:

- Connectors must not cross.
- All words in a sentence have to be attached to the tree.

Valid sentence: All local and global rules can be applied.

Rules

Global Rules:

- Connectors must not cross.
- All words in a sentence have to be attached to the tree.

Valid sentence: All local and global rules can be applied.

Outline

Notation

Connectors:

- +: connector to the right
- -: connector to the left

Combining connectors:

- &: Combines two connectors, both must be connected
- or: Combines two connectors, at least one must be connected
- {}: Connectors in curly brackets are optional.
- @: Connectors with an @ must be connected at least once but can have multiple instances.

Notation

Connectors:

- +: connector to the right
- -: connector to the left

Combining connectors:

- &: Combines two connectors, both must be connected
- or: Combines two connectors, at least one must be connected
- {}: Connectors in curly brackets are optional.
- @: Connectors with an @ must be connected at least once but can have multiple instances.

Notation

Connectors:

- +: connector to the right
- -: connector to the left

Combining connectors:

- &: Combines two connectors, both must be connected
- or: Combines two connectors, at least one must be connected
- {}: Connectors in curly brackets are optional.
- @: Connectors with an @ must be connected at least once but can have multiple instances.

Notation

Connectors:

- +: connector to the right
- -: connector to the left

Combining connectors:

- &: Combines two connectors, both must be connected
- or: Combines two connectors, at least one must be connected
- {}: Connectors in curly brackets are optional.
- @: Connectors with an @ must be connected at least once but can have multiple instances.

Notation

Connectors:

- `+`: connector to the right
- `-`: connector to the left

Combining connectors:

- `&`: Combines two connectors, both must be connected
- `or`: Combines two connectors, at least one must be connected
- `{}`: Connectors in curly brackets are optional.
- `@`: Connectors with an `@` must be connected at least once but can have multiple instances.

Notation

Connectors:

- `+`: connector to the right
- `-`: connector to the left

Combining connectors:

- `&`: Combines two connectors, both must be connected
- `or`: Combines two connectors, at least one must be connected
- `{}`: Connectors in curly brackets are optional.
- `@`: Connectors with an `@` must be connected at least once but can have multiple instances.

Notation

Connectors:

- `+`: connector to the right
- `-`: connector to the left

Combining connectors:

- `&`: Combines two connectors, both must be connected
- `or`: Combines two connectors, at least one must be connected
- `{}`: Connectors in curly brackets are optional.
- `@`: Connectors with an `@` must be connected at least once but can have multiple instances.

Examples

Notation:

- word: A+;
- word: A+ & B-;
- word: (A+ or B-) & ((C- & A+ & (D- or E-)) or F+);
- word: (A+ or B+) & {C- & (D+ or E-)} & {@F+};

Examples

Notation:

- word: A+;
- word: A+ & B-;
- word: (A+ or B-) & ((C- & A+ & (D- or E-)) or F+);
- word: (A+ or B+) & {C- & (D+ or E-)} & {@F+};

Examples

Notation:

- word: A+;
- word: A+ & B-;
- word: (A+ or B-) & ((C- & A+ & (D- or E-)) or F+);
- word: (A+ or B+) & {C- & (D+ or E-)} & {@F+};

Examples

Notation:

- word: A+;
- word: A+ & B-;
- word: (A+ or B-) & ((C- & A+ & (D- or E-)) or F+);
- word: (A+ or B+) & {C- & (D+ or E-)} & {@F+};

Examples

Notation:

- word: A+;
- word: A+ & B-;
- word: (A+ or B-) & ((C- & A+ & (D- or E-)) or F+);
- word: (A+ or B+) & {C- & (D+ or E-)} & {@F+};

Outline

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.

Conclusion

Dependency and Constituent Parsing don't just seem to be similar.

- They are strongly equivalent.
- Constituency Grammar has to be limited.
  - Single word has to be designated as head
  - Phrase must not have designation or name apart from head
- Constituency Grammars currently in use are notational variants of Dependency Grammars.

Advantages of Dependency / Link Grammars:

- Dependency Parsing approach is similar to human reading
- Link Parsing trees can be converted into Dependency and Constituency trees.