Lecture 3: Writers' aids: Grammatical errors

LING-351 Language Technology and LLMs

Instructor: Hakyung Sung

September 2, 2025

*Acknowledgment: These course slides are based on materials by Lelia Glass @ Georgia Tech (Course: Language & Computers)

Table of contents

Review

Spelling checker

Traditional method: Dictionary + Edit Distance

- Relies on a **dictionary** of correct words, built from a corpus

Spelling checker

Traditional method: Dictionary + Edit Distance

- Relies on a **dictionary** of correct words, built from a corpus
- Calculates distance between misspelling and candidates (very simple, but works quite well)

Spelling checker

Traditional method: Dictionary + Edit Distance

- Relies on a **dictionary** of correct words, built from a corpus
- Calculates distance between misspelling and candidates (very simple, but works quite well)
- Suggests the closest candidate as the correction

Spelling checker

Traditional method: Dictionary + Edit Distance

- Relies on a **dictionary** of correct words, built from a corpus
- Calculates distance between misspelling and candidates (very simple, but works quite well)
- Suggests the closest candidate as the correction
- Adds some weights for more realistic correction

Why context matters in spell-checking

Example: Someone types:

`You put the catt before the horse.`

- `put the cart before the horse`
- `put the cat before the horse`

Using N-grams to model context

N-grams are sequences of *n* elements (e.g., words or characters):

- **Unigram** = one word: `the`

Using N-grams to model context

N-grams are sequences of *n* elements (e.g., words or characters):

- **Unigram** = one word: `the`
- **Bigram** = two-word sequence: `the cat`

Using N-grams to model context

N-grams are sequences of *n* elements (e.g., words or characters):

- **Unigram** = one word: `the`
- **Bigram** = two-word sequence: `the cat`
- **Trigram** = three-word sequence: `put the cat`

Other approaches

- **Statistical Language Models (n-grams)** Use probability of surrounding context e.g., *I went to the shcool* → “school” is more probable

Other approaches

- **Statistical Language Models (n-grams)** Use probability of surrounding context e.g., *I went to the shcool* → “school” is more probable
- **Neural Spell Checkers (Deep Learning)** Seq2Seq / Transformer-based models generate corrected text Examples: ChatGPT, Grammarly, Google Docs
- **Hybrid Approaches** Combine edit distance with language models; pick the highest probability candidate

How common are spelling errors?

- About 2–3% of all typed words on a full-size keyboard are misspelled by proficient adults (Flor et al., 2015)

Table 2. Summary statistics for the ETS Spelling Corpus

	GRE Argument	GRE Issue	TOEFL Independent	TOEFL Integrated	TOTAL
Total essays	750	750	750	750	3,000
Essays without misspellings	60	21	18	21	120
Total Word Count	263,578	336,301	212,930	151,031	963,840
Average Word Count	351	448	284	201	321
Total count of Misspellings	5,935	7,962	7,285	5,230	26,412
Misspellings as % of all words	2.25%	2.37%	3.42%	3.46%	2.74%

Figure 1: Flor et al. (2015), p. 112

How common are spelling errors?

- About 2–3% of all typed words on a full-size keyboard are misspelled by proficient adults (Flor et al., 2015)

Table 2. Summary statistics for the ETS Spelling Corpus

	GRE Argument	GRE Issue	TOEFL Independent	TOEFL Integrated	TOTAL
Total essays	750	750	750	750	3,000
Essays without misspellings	60	21	18	21	120
Total Word Count	263,578	336,301	212,930	151,031	963,840
Average Word Count	351	448	284	201	321
Total count of Misspellings	5,935	7,962	7,285	5,230	26,412
Misspellings as % of all words	2.25%	2.37%	3.42%	3.46%	2.74%

Figure 1: Flor et al. (2015), p. 112

- On a mobile phone, however, about 40% of words are misspelled (Grammarly, 2019)

How common are spelling errors?

- About 2–3% of all typed words on a full-size keyboard are misspelled by proficient adults (Flor et al., 2015)

Table 2. Summary statistics for the ETS Spelling Corpus

	GRE Argument	GRE Issue	TOEFL Independent	TOEFL Integrated	TOTAL
Total essays	750	750	750	750	3,000
Essays without misspellings	60	21	18	21	120
Total Word Count	263,578	336,301	212,930	151,031	963,840
Average Word Count	351	448	284	201	321
Total count of Misspellings	5,935	7,962	7,285	5,230	26,412
Misspellings as % of all words	2.25%	2.37%	3.42%	3.46%	2.74%

Figure 1: Flor et al. (2015), p. 112

- On a mobile phone, however, about 40% of words are misspelled (Grammarly, 2019)
- More multi-error misspellings and real-word errors due to auto-complete (e.g., ***restaurant*** → typed as ***restuarnt*** → auto-corrected to ***restart***)

Lesson plan

Lesson plan

- ~~Review~~
- Grammar

Lesson plan

- ~~Review~~
- Grammar
- Part of Speech (POS)

Lesson plan

- ~~Review~~
- Grammar
- Part of Speech (POS)
- Dependency Grammar

Lesson plan

- ~~Review~~
- Grammar
- Part of Speech (POS)
- Dependency Grammar
- Grammar Checker

Lesson plan

- ~~Review~~
- Grammar
- Part of Speech (POS)
- Dependency Grammar
- Grammar Checker
- Wrap-up

Key idea: Building a grammar checker begins with understanding key linguistic categories

Grammar

Two viewpoints on grammar

- **Descriptive Grammar** Describes how people actually use language in real life. (Focus: *what speakers do*)

Two viewpoints on grammar

- **Descriptive Grammar** Describes how people actually use language in real life. (Focus: *what speakers do*)
- **Prescriptive Grammar** Lays down rules for how language *should* be used. (Focus: *what speakers ought to do*)

Examples of grammar in use

- *“I ain’t got no books”* → Descriptive grammar may accept this in some English varieties.

Examples of grammar in use

- *“I ain’t got no books”* → Descriptive grammar may accept this in some English varieties.
- **“Book the read student”* → Descriptive grammar rejects this as ungrammatical (not used by any speaker).

Examples of grammar in use

- *"I ain't got no books"* → Descriptive grammar may accept this in some English varieties.
- **"Book the read student"* → Descriptive grammar rejects this as ungrammatical (not used by any speaker).
- Prescriptive grammar?

Grammar checkers and mixed rules

- Commercial grammar checkers may apply both descriptive rules and prescriptive rules.

Questions (*Shared deck*)

1. Do you think grammar checkers should allow forms (e.g., *“gonna”*, *“ain’t”*)? Why or why not?

Grammar checkers and mixed rules

- Commercial grammar checkers may apply both descriptive rules and prescriptive rules.

Questions (*Shared deck*)

1. Do you think grammar checkers should allow forms (e.g., *“gonna”*, *“ain’t”*)? Why or why not?
2. Do prescriptive corrections (e.g., “less people” → “fewer people”; “He suggested me to go” → “He suggested that I go”) really improve clarity, or just follow rules?

Grammar checkers and mixed rules

- Commercial grammar checkers may apply both descriptive rules and prescriptive rules.

Questions (*Shared deck*)

1. Do you think grammar checkers should allow forms (e.g., *"gonna"*, *"ain't"*)? Why or why not?
2. Do prescriptive corrections (e.g., "less people" → "fewer people"; "He suggested me to go" → "He suggested that I go") really improve clarity, or just follow rules?
3. **Even without a grammar checker, speakers often recognize what "sounds right." How do you think this internal system of grammar works in your mind?**

Linguistics and syntax

- In Linguistics, grammar is often studied under syntax.

Linguistics and syntax

- In Linguistics, grammar is often studied under syntax.
- Two key concepts for grammar checkers:

Linguistics and syntax

- In Linguistics, grammar is often studied under syntax.
- Two key concepts for grammar checkers:
  - **Part of Speech (POS)** – classifies each word

Linguistics and syntax

- In Linguistics, grammar is often studied under syntax.
- Two key concepts for grammar checkers:
  - **Part of Speech (POS)** – classifies each word
  - **Dependency grammar** – shows how words are connected

Linguistics and syntax

- In Linguistics, grammar is often studied under syntax.
- Two key concepts for grammar checkers:
  - **Part of Speech (POS)** – classifies each word
  - **Dependency grammar** – shows how words are connected
- Why this matters for grammar checkers?

Part of Speech (POS)

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government
  - **V___**: play, run, believe

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government
  - **V___**: play, run, believe
  - **A___**: fun, beautiful

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government
  - **V___**: play, run, believe
  - **A___**: fun, beautiful
  - **A___**: well, heavily

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N**___: reindeer, game, government
  - **V**___: play, run, believe
  - **A**___: fun, beautiful
  - **A**___: well, heavily
  - **P**___: on, into

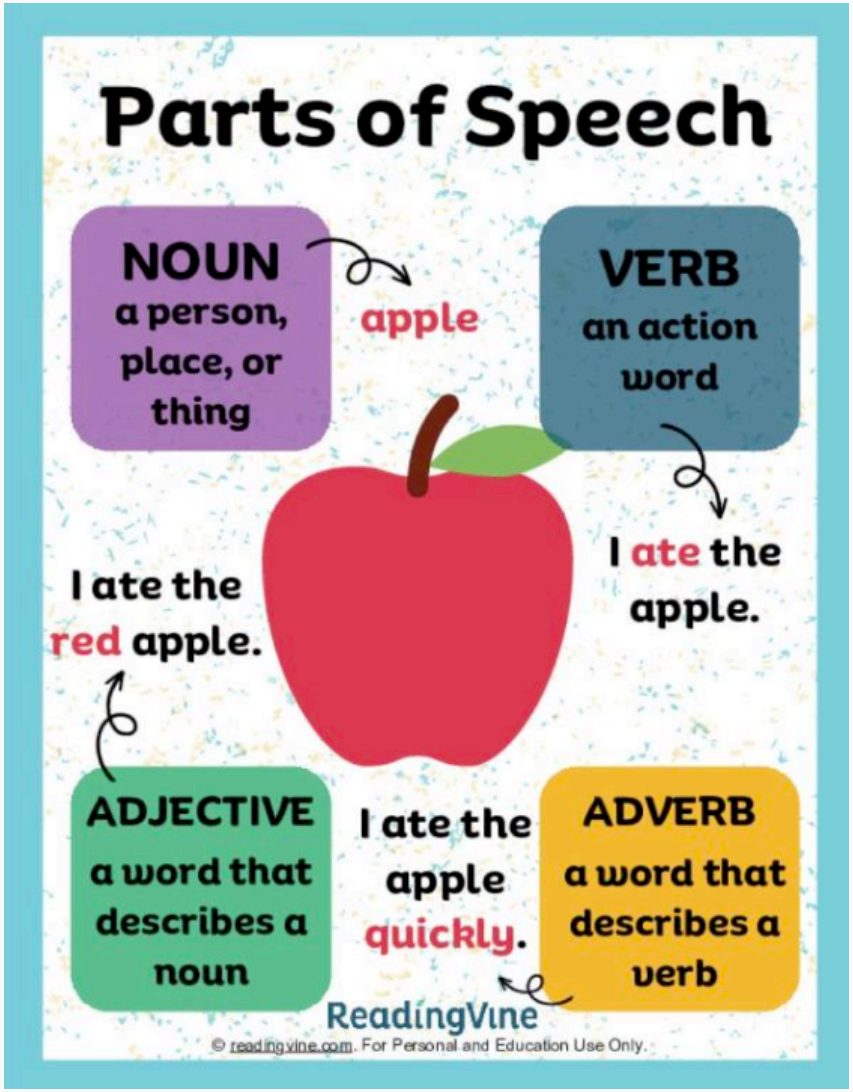
Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government
  - **V___**: play, run, believe
  - **A___**: fun, beautiful
  - **A___**: well, heavily
  - **P___**: on, into
  - **A___/D___**: a, the, some

Part-of-Speech (POS)

- A word's POS determines how it fits into a sentence.
- POS is also called **lexical category**.
- Main POS categories:
  - **N___**: reindeer, game, government
  - **V___**: play, run, believe
  - **A___**: fun, beautiful
  - **A___**: well, heavily
  - **P___**: on, into
  - **A___/D___**: a, the, some
  - **C___**: and, or

Part-of-Speech (POS)



How we identify POS

- Examples:

How we identify POS

- Examples:
  - *This car is very interesting.*

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?
  - What patterns do you see in how these words are used?

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?
  - What patterns do you see in how these words are used?
  - How could these patterns help us decide the POS of a word?

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?
  - What patterns do you see in how these words are used?
  - How could these patterns help us decide the POS of a word?
- We usually identify POS by:

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?
  - What patterns do you see in how these words are used?
  - How could these patterns help us decide the POS of a word?
- We usually identify POS by:
  - **Distribution:** where a word appears in a sentence (e.g., nouns after articles, verbs after subjects)

How we identify POS

- Examples:
  - *This car is very interesting.*
  - *This car mooked fast.*
  - *This nony car mooked fast.*
- How we identify lexical category?
  - What patterns do you see in how these words are used?
  - How could these patterns help us decide the POS of a word?
- We usually identify POS by:
  - **Distribution:** where a word appears in a sentence (e.g., nouns after articles, verbs after subjects)
  - **Morphology:** how a word changes form (e.g., verbs mark tense: *play* → *played*, sometimes irregularly: *go* → *went*)

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns
- **Morphology / Tense** as features

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns
- **Morphology / Tense** as features
  - Inflectional endings: *-ed*, *-ing*, *-s*

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns
- **Morphology / Tense** as features
  - Inflectional endings: *-ed*, *-ing*, *-s*
  - Irregular verb forms: *go* → *went*, *take* → *took*

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns
- **Morphology / Tense** as features
  - Inflectional endings: *-ed*, *-ing*, *-s*
  - Irregular verb forms: *go* → *went*, *take* → *took*
  - Used as binary/frequency features in classifiers

Notes. POS tagging: Traditional NLP

- **Part of Speech (POS)** as features
  - POS tags/frequency used in text classification (e.g., proportion of nouns, verbs, adjectives)
  - POS n-grams (e.g., DET + NOUN + VERB) to capture patterns
- **Morphology / Tense** as features
  - Inflectional endings: *-ed*, *-ing*, *-s*
  - Irregular verb forms: *go* → *went*, *take* → *took*
  - Used as binary/frequency features in classifiers
- Before deep learning, POS and morphology were essential **hand-crafted features** (based on the prescribed rules).

Notes. POS tagging: Current NLP


```
# sent_id = file00321.txt_41
# text = And there are a big amusement park
1   And _   CCONJ   CC  _   3   cc  _   _
2   there   _   PRON    EX  _   3   expl    _   _
3   are _   VERB    VBP _   0   root    _   _
4   a   _   DET DT  _   7   det _   _
5   big _   ADJ JJ  _   7   amod    _   _
6   amusement   _   NOUN    NN  _   7   compound    _   _
7   park    _   NOUN    NN  _   3   nsubj   _   _
```

From words to phrases

- Words combine into constituents based on POS:

From words to phrases

- Words combine into constituents based on POS:
  - **the reindeer** = article + noun = noun phrase

From words to phrases

- Words combine into constituents based on POS:
  - **the reindeer** = article + noun = noun phrase
  - **play games** = verb + noun phrase = verb phrase

From words to phrases

- Words combine into constituents based on POS:
  - **the reindeer** = article + noun = noun phrase
  - **play games** = verb + noun phrase = verb phrase
- Constituents combine based on **phrasal category**:

From words to phrases

- Words combine into constituents based on POS:
  - **the reindeer** = article + noun = noun phrase
  - **play games** = verb + noun phrase = verb phrase
- Constituents combine based on **phrasal category**:
  - **Noun Phrase** + **Verb Phrase** = Sentence

Structure over meaning

- Chomsky (1957): *“Colorless green ideas sleep furiously”*

Structure over meaning

- Chomsky (1957): *“Colorless green ideas sleep furiously”*
- Nonsensical meaning, but:

Structure over meaning

- Chomsky (1957): *“Colorless green ideas sleep furiously”*
- Nonsensical meaning, but:
  - Correct lexical and phrasal categories

Structure over meaning

- Chomsky (1957): *“Colorless green ideas sleep furiously”*
- Nonsensical meaning, but:
  - Correct lexical and phrasal categories
  - Grammatically well-formed

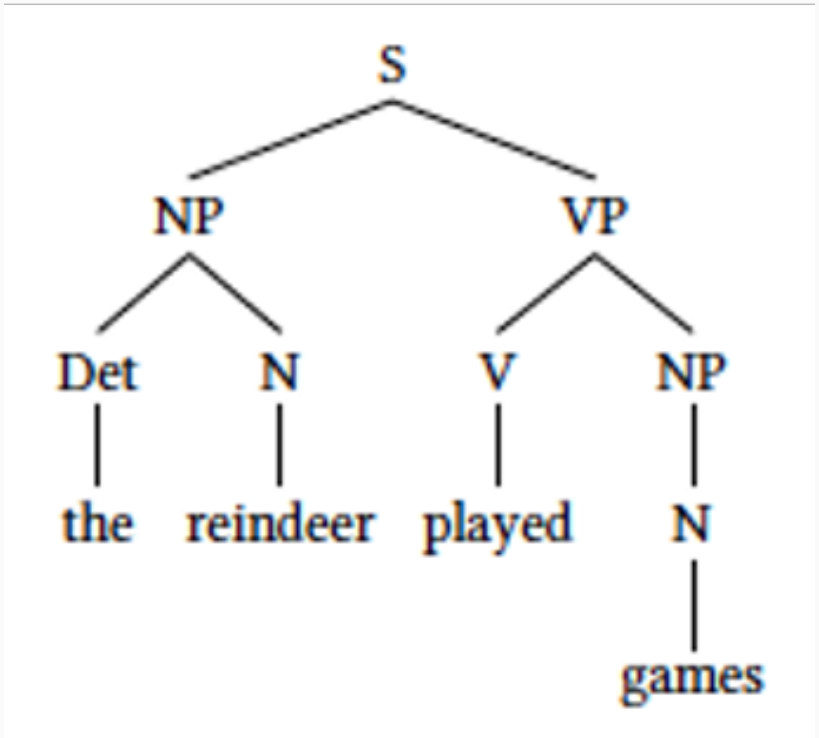
Structure over meaning

- Chomsky (1957): *“Colorless green ideas sleep furiously”*
- Nonsensical meaning, but:
  - Correct lexical and phrasal categories
  - Grammatically well-formed
- Syntax is about **structure**, not always meaning.

Phrase structure grammar (Chomsky, 1950s-1960s)

Lexicon:
N → *reindeer, dragon, lunch, game, evening, morning*
V(trans) → *play, eat*
V(intrans) → *run, swim, dance*
Adj → *fun, beautiful, interesting*
Det → *the, a, some, many*
P → *for, in, to, at*

Phrase structure rules:
S → NP VP
VP → V(trans) NP
VP → V(intrans)
NP → Det (A*) N
NP → N
NP → NP PP
PP → P NP



Frameworks for analyzing syntax

- Linguists formalize sentence structure using grammar frameworks:

Frameworks for analyzing syntax

- Linguists formalize sentence structure using grammar frameworks:
  - Phrase Structure Grammar (common in **linguistics**)

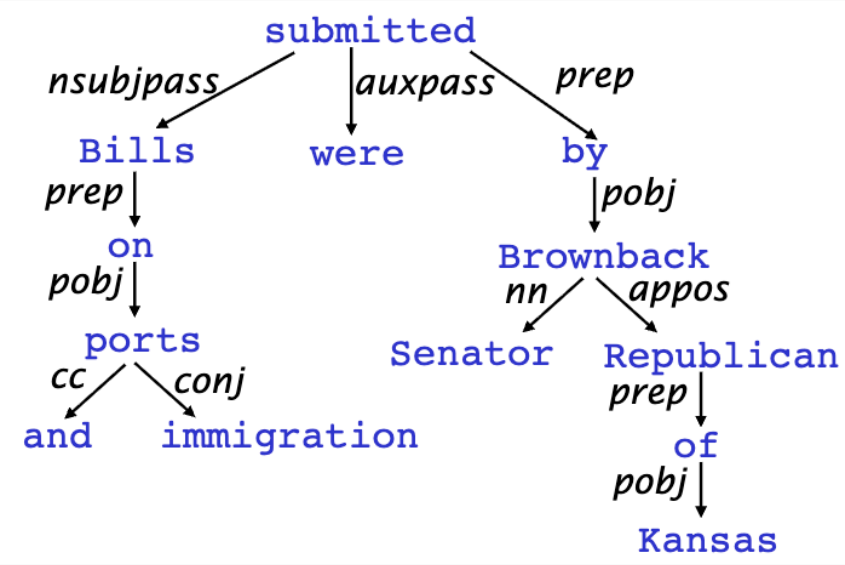
Frameworks for analyzing syntax

- Linguists formalize sentence structure using grammar frameworks:
  - Phrase Structure Grammar (common in **linguistics**)
  - **Dependency Grammar** (widely used in **NLP**)

Dependency grammar

What is dependency grammar?

Dependency syntax postulates that syntactic structure consists of **relationships** between lexical items, normally binary asymmetric relations ("arrows") called dependencies.



Sourced from:
https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1162/handouts/SLoSP-2014-4-dependencies.pdf

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)
  - `dog` → dependent with relation `nsubj`

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)
  - `dog` → dependent with relation `nsubj`
  - `cat` → dependent with relation `obj`

What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)
  - `dog` → dependent with relation `nsubj`
  - `cat` → dependent with relation `obj`
  - `The` → dependent of both nouns with relation `det`

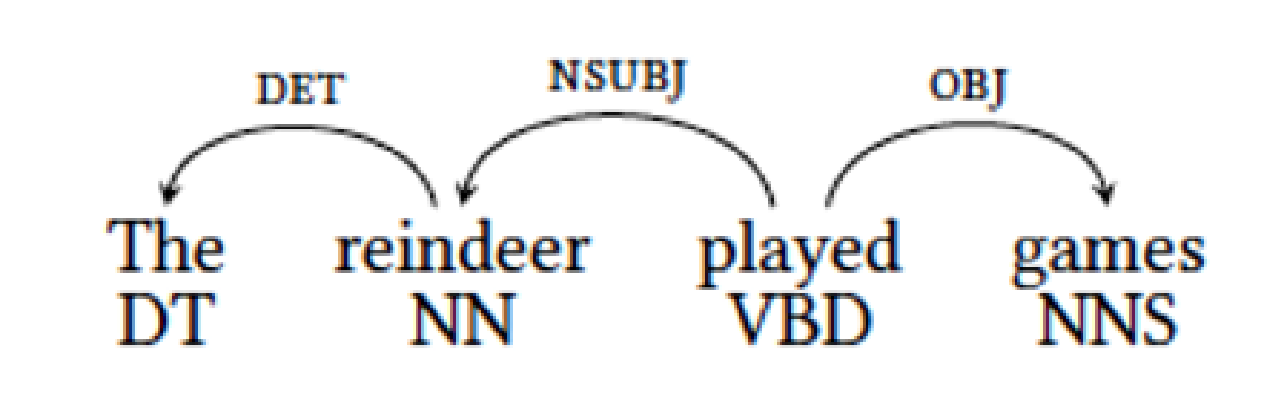
What is dependency grammar?

- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)
  - `dog` → dependent with relation `nsubj`
  - `cat` → dependent with relation `obj`
  - `The` → dependent of both nouns with relation `det`
  - `.` → dependent with relation `punct`

What is dependency grammar?

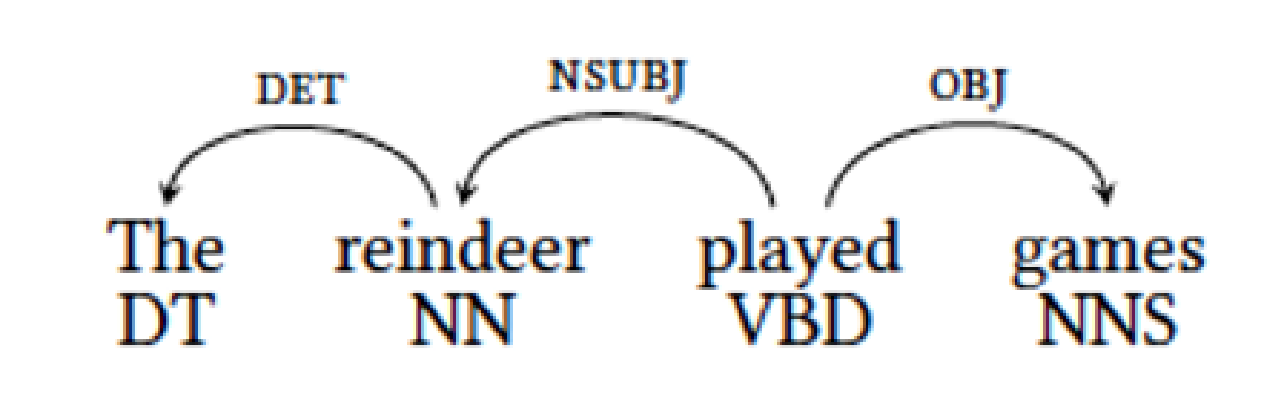
- Represents syntax as binary, asymmetric relations between words.
  - One word is the **head**, and the other is the **dependent**.
  - Unlike phrase structure grammar, it does not group words into large phrases — **it focuses on direct word-to-word links**.
- Each dependency relation is **typed**:
  - `nsubj` = nominal subject
  - `obj` = object
  - `det` = determiner
- Example: *The dog chased the cat.*
  - `chased` → head verb (ROOT)
  - `dog` → dependent with relation `nsubj`
  - `cat` → dependent with relation `obj`
  - `The` → dependent of both nouns with relation `det`
  - `.` → dependent with relation `punct`
- Practice: *The reindeer played games.*

Example: *The reindeer played games*



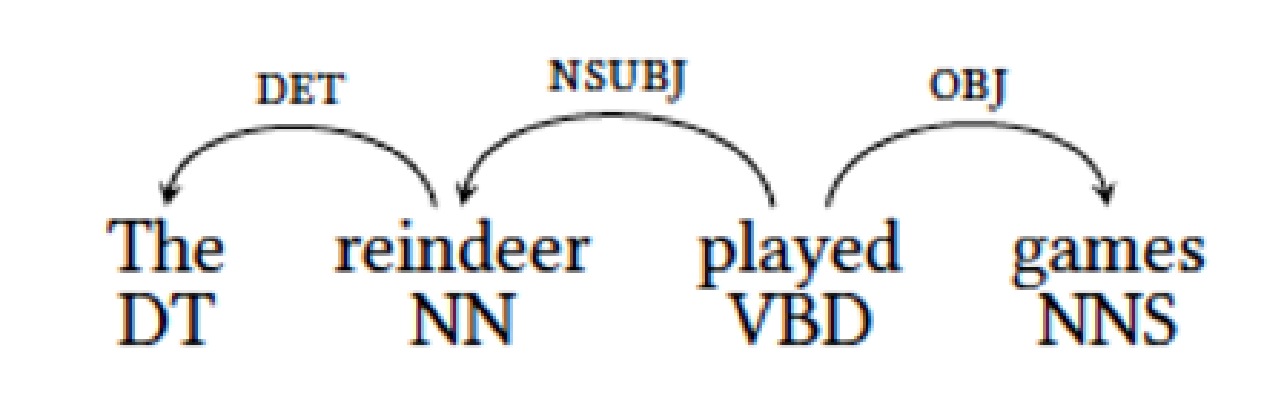
- *the* → dependent of *reindeer* via `det`

Example: *The reindeer played games*



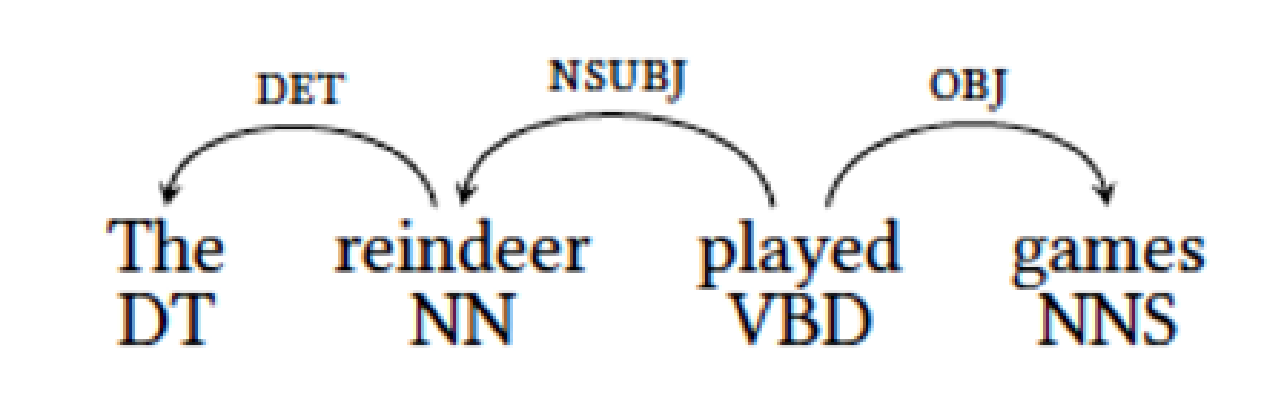
- *the* → dependent of *reindeer* via `det`
- *reindeer* → subject of *played* via `nsubj`

Example: *The reindeer played games*



- *the* → dependent of *reindeer* via `det`
- *reindeer* → subject of *played* via `nsubj`
- *games* → object of *played* via `obj`

Example: *The reindeer played games*



- *the* → dependent of *reindeer* via `det`
- *reindeer* → subject of *played* via `nsubj`
- *games* → object of *played* via `obj`
- *played* = root of the sentence

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**
  - Free word-order languages (e.g., Korean, Russian)

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**
  - Free word-order languages (e.g., Korean, Russian)
  - Long-distance dependencies (e.g., *What did you eat?*)

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**
  - Free word-order languages (e.g., Korean, Russian)
  - Long-distance dependencies (e.g., *What did you eat?*)
- **Practical impact:**

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**
  - Free word-order languages (e.g., Korean, Russian)
  - Long-distance dependencies (e.g., *What did you eat?*)
- **Practical impact:**
  - Widely adopted in open-source NLP libraries (e.g., spaCy, Stanza, UDPipe)

Why use dependency grammar?

- **Cross-linguistic:** works across languages, not tied to a fixed word order.
- **Handles complexity:**
  - Free word-order languages (e.g., Korean, Russian)
  - Long-distance dependencies (e.g., *What did you eat?*)
- **Practical impact:**
  - Widely adopted in open-source NLP libraries (e.g., spaCy, Stanza, UDPipe)
  - State-of-the-art parsers trained on this format work very effectively in many languages (`https://stanfordnlp.github.io/stanza/performance.html`)

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- **What is UD?**

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies
  - Open community effort: 600+ contributors, 200+ treebanks, 150+ languages (`https://universaldependencies.org/`)

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- **What is UD?**
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies
  - Open community effort: 600+ contributors, 200+ treebanks, 150+ languages (`https://universaldependencies.org/`)
- **Why it matters:**

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies
  - Open community effort: 600+ contributors, 200+ treebanks, 150+ languages (`https://universaldependencies.org/`)
- Why it matters:
  - Enables cross-linguistic comparison

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies
  - Open community effort: 600+ contributors, 200+ treebanks, 150+ languages (`https://universaldependencies.org/`)
- Why it matters:
  - Enables cross-linguistic comparison
  - Supports language typology research

Universal Dependencies (UD) Project

- A large multilingual *corpus* annotated in a consistent dependency format.
- What is UD?
  - A framework for consistent annotation of grammar across languages
  - Covers: parts of speech, morphological features, syntactic dependencies
  - Open community effort: 600+ contributors, 200+ treebanks, 150+ languages (`https://universaldependencies.org/`)
- Why it matters:
  - Enables cross-linguistic comparison
  - Supports language typology research
  - Provides a foundation for multilingual NLP tools

More guides

The secret to understanding the design and current success of UD is to realize that the design is a very subtle compromise between approximately 6 things:

1. UD needs to be satisfactory on linguistic analysis grounds for **individual** languages.
2. UD needs to be good for linguistic **typology**, i.e., providing a suitable basis for bringing out cross-linguistic parallelism across languages and language families.
3. UD must be suitable for rapid, consistent **annotation** by a human annotator.
4. UD must be suitable for computer **parsing** with high accuracy.
5. UD must be easily comprehended and used by a **non-linguist**, whether a language learner or an engineer with prosaic needs for language processing. We refer to this as seeking a habitable design, and it leads us to favor traditional grammar notions and terminology.
6. UD must support well downstream language **understanding** tasks (relation extraction, reading comprehension, machine translation, …).

Sourced from: `https://people.cs.georgetown.edu/nschneid/p/UD-for-English.pdf`

Parsing with dependency grammar

- Goal: Automatically generate a tree for a new sentence

Let's play: `https://demos.explosion.ai/displacy`

Parsing with dependency grammar

- Goal: Automatically generate a tree for a new sentence
- Steps:

Let's play: `https://demos.explosion.ai/displacy`

Parsing with dependency grammar

- Goal: Automatically generate a tree for a new sentence
- Steps:
    1. Tag words with part of speech

Let's play: `https://demos.explosion.ai/displacy`

Parsing with dependency grammar

- Goal: Automatically generate a tree for a new sentence
- Steps:
  1. Tag words with part of speech
  2. Assign dependency relations

Let's play: `https://demos.explosion.ai/displacy`

Parsing with dependency grammar

- Goal: Automatically generate a tree for a new sentence
- Steps:
  1. Tag words with part of speech
  2. Assign dependency relations
- Built using machine learning and large annotated corpora

Let's play: `https://demos.explosion.ai/displacy`

Grammar checker

From grammar to grammar Checkers

- So far, we've built a foundation by analyzing sentence structure.
- Now we can apply this knowledge to automatic grammar checking. For example:
- Try to assign a **dependency parse**.
  - If parsing fails → likely an error.
  - If parsing succeeds → compare to known grammar rules.

Rule-Based error detection

- Use **hand-written rules** (based on [descriptive/prescriptive] grammatical knowledge) to detect common errors.

Rule-Based error detection

- Use **hand-written rules** (based on [descriptive/prescriptive] grammatical knowledge) to detect common errors.
- Example:

Rule-Based error detection

- Use **hand-written rules** (based on [descriptive/prescriptive] grammatical knowledge) to detect common errors.
- Example:
  - If the subject (`nsubj`) is tagged `NN` (singular noun),

Rule-Based error detection

- Use **hand-written rules** (based on [descriptive/prescriptive] grammatical knowledge) to detect common errors.
- Example:
  - If the subject (`nsubj`) is tagged `NN` (singular noun),
  - Then the verb should be tagged `VBZ` (3rd-person singular).

Rule-Based error detection

- Use **hand-written rules** (based on [descriptive/prescriptive] grammatical knowledge) to detect common errors.
- Example:
  - If the subject (`nsubj`) is tagged `NN` (singular noun),
  - Then the verb should be tagged `VBZ` (3rd-person singular).
  - *The dog swim* (X) → should be *The dog swims*.

What you need to build a grammar checker

- A fast and accurate **dependency parser**?

What you need to build a grammar checker

- A fast and accurate **dependency parser**?
- A set of **hand-written grammar rules**?

What you need to build a grammar checker

- A fast and accurate **dependency parser**?
- A set of **hand-written grammar rules**?
- Confusion sets for **commonly misused words**?

What you need to build a grammar checker

- A fast and accurate **dependency parser**?
- A set of **hand-written grammar rules**?
- Confusion sets for **commonly misused words**?
- What else?

Rule-Based vs. LLM-Based Grammar Checkers

Rule-Based Checkers

- Use explicit grammar rules and POS/dependency tags
- Rely on parsing + handcrafted logic
- Example: Check for subject–verb agreement via `nsubj` and `VBZ`
- Explainable and controllable, but less flexible

LLM-Based Checkers

- Use large neural language models (e.g., GPT, BERT)
- *Learn grammar implicitly from vast corpora*
- Can handle diverse errors *without explicit rules*
- Often produce fluent rewrites, but less transparent

Wrap-up

Wrap-up

- In Linguistics, grammar is often studied under syntax.
- Two key concepts for grammar checkers:
  - **Part of Speech (POS)** – classifies each word
  - **Dependency grammar** – shows how words are connected
- **Why this matters for grammar checkers?**
  - Detect whether words fit together according to rules
  - Spot unusual or incorrect structures

Updates

Syllabus updated

The syllabus has been updated:

2	9/2	Writer's aids: Grammar errors	[LC] Ch.2.5-2.8	
	9/4	Python tutorial 1		Exercise 1
3	9/9	Computer-assisted language learning	[LC] Ch.3	
	9/11	Python tutorial 2		Exercise 2
4	9/16	Text as data		
	9/18	Python tutorial 3		
5	9/23	Word vectors		
	9/25	Python tutorial 4		Exercise 3

So, bring your laptop on Thursday!