Q7) Compare dependency vs constituency grammars—representation & parsing complexity. NLP_Assignment No. 2

Representation

- **Constituency (phrase-structure):** sentences form a **hierarchical tree** of phrases (NP, VP, PP…). Good for capturing **phrase boundaries**, **subcategorization frames**, and **ellipsis/coordination** structures.
- **Dependency:** a **directed tree** over words where each token (except ROOT) has exactly one head; edges are labeled with grammatical relations (e.g., `nsubj`, `obj`, `amod`). Compact, language-agnostic, and directly supports **predicate–argument** extraction.

Typical outputs

- Constituency: `S → NP VP`, `NP → Det N` … (bracketed parse/tree).
- Dependency: `nsubj(sleeps, cat)`, `det(cat, the)`.

Parsing complexity (classic algorithms)

- **Constituency:** CKY for CNF **$O(n^3 \cdot |G|)$**; lexicalized parsers often retain cubic-ish behavior (with pruning).
- **Dependency:**
  - **Projective** dynamic programming (Eisner): **$O(n^3)$** decoding.
  - **Non-projective** MST (Chu–Liu/Edmonds): **$O(n^2)$** (or **$O(n^2 \log n)$** with certain implementations).
  - **Transition-based (greedy/beam)**: **$O(n)$** (greedy) to **$O(b \cdot n)$** (beam width *b*).

When to use which

- **Constituency** shines for **text simplification**, **question formation**, **semantic role labeling** that benefits from phrasal spans.
- **Dependency** excels in **IE/KB population**, **relation extraction**, and **cross-lingual parsing** (fewer assumptions about phrase structure).

Q8) Limitations of transition-based parsing for long-range dependencies. NLP_Assignment No. 2

1. **Greedy error propagation.** Early wrong SHIFT/REDUCE causes **downstream irreversible mistakes**, especially when the head is far away.
2. **Limited horizon.** Classic feature sets look at **top of stack / front of buffer**; even with BiLSTM/Transformer encoders, the **decision space** is still local and can miss **distant cues** (e.g., subject–verb agreement across long relative clauses).
3. **Beam search trade-off.** Wider beams help but increase **latency**; still not globally optimal.
4. **Non-projectivity.** Standard arc-eager/arc-standard require extra transitions (e.g., **SWAP**) to handle crossing arcs—adds complexity and can hurt stability on long-distance, non-projective links.
5. **Exposure bias.** Trained on gold transitions; at test time, model sees its own **errorful states**, degrading performance on long dependencies.

Common mitigations

- **Beam search**, **dynamic oracles**, **globally normalized objectives** (CRF/structured losses).

- **Pre-pruning with taggers** or **pointer-network heads** to bias toward plausible long arcs.
- Adding **SWAP** (or similar) transitions for non-projectivity.

Q9) MST-based vs transition-based dependency parsing—strengths & weaknesses. NLP_Assignment No. 2

MST-based (graph-based)

- **Idea:** score all possible head–dependent arcs; decode the **maximum spanning tree**.
- **Pros**
  - **Global optimum** under the scoring model → robust on **long-range** dependencies.
  - Handles **non-projective** structures natively (Chu–Liu/Edmonds).
  - Naturally incorporates **pairwise global features** (and with higher-order variants, siblings/grandparents).
- **Cons**
  - Decoding cost (typically **$O(n^2)$** arcs; higher-order decoding can be slower).
  - Historically less “incremental” (harder to use in streaming/interactive settings).
  - Requires careful **regularization** to avoid overfitting on dense arc scores.

Transition-based

- **Idea:** build the tree incrementally with SHIFT/ARC/REDUCE actions.
- **Pros**
  - **Very fast** (greedy **$O(n)$**), **low latency**; great for on-device or high-throughput pipelines.
  - Easy to add **rich contextual encodings** into action classification.
  - Naturally **incremental** (useful for simultaneous ASR→parse or interactive systems).
- **Cons**
  - **Error propagation**; weaker on **long-distance** arcs.
  - Needs beam/global normalization to rival graph-based accuracy, which reduces speed gains.

Rule of thumb

- **High accuracy / long sentences / non-projectivity → MST/graph-based**.
- **Real-time / edge devices / streaming → transition-based** (possibly with a small beam).

Q10) Why do we need probabilistic models (PCFGs) for syntactic ambiguity vs deterministic approaches? NLP_Assignment No. 2

Ambiguity is pervasive.
Consider **PP-attachment**: *“I saw the man with a telescope.”*
Two parses: `(saw [the man [with a telescope]])` vs `([saw …] [with a telescope])`.

Deterministic parsers (fixed precedence/hand-written rules) can:

- Be **brittle**: one rule rarely fits all domains.

- Fail on **lexical idiosyncrasies** and **domain shifts**.

PCFG advantage

- Assigns **probabilities to rules**. If the corpus shows that `PP` more often attaches to `NP` than `VP` in a given context, the model **prefers the more likely parse**.
- **Learned from data** (treebanks) → adapts to **genre/domain**.
- Extensible to **lexicalization** and **neuralized** PCFGs (context-sensitive probabilities).

Mini-example (toy numbers)

- `P(VP→VP PP)=0.2`, `P(NP→NP PP)=0.6`.
  The parse attaching `PP` to `NP` will typically have a **higher overall probability**, resolving the ambiguity **quantitatively**, not by brittle heuristics.

Q11) Design a hybrid parser: combine PCFG probabilities with dependency techniques for better accuracy. NLP_Assignment No. 2

Goal: marry **phrasal** strengths (PCFG) with **head–dependent** precision (dependency).

Architecture (one effective pattern)

1. **k-best PCFG stage (coarse-to-fine).**
   - Parse with a PCFG (neuralized if available) using coarse→fine grammars to get **k best constituency trees** with probabilities.
2. **Constituency→dependency projection.**
   - Convert each candidate to a dependency tree (head rules).
3. **Graph-based re-ranker.**
   - Score each candidate's dependency arcs using a **biaffine graph scorer** (or MST decoder if we relax to arbitrary arcs).
   - Combine scores:

$$\text{Score}(T) \;=\; \lambda \cdot \log P_{\text{PCFG}}(T) \;+\; (1-\lambda) \cdot \sum_{(h\to d)\in T} s_{\text{dep}}(h,d)$$

   - Choose the best-scoring tree (or decode a fresh MST constrained by phrasal spans).
4. **Global constraints.**
   - Enforce **phrase boundaries** (from k-best) while **optimizing dependencies** → reduces illegal structures and improves label accuracy.
5. **Training.**
   - **Multi-task**: share encoders; losses for **constituent spans** and **dependency arcs**. Tune λ on dev data.

Why this works

- PCFG supplies **global phrasal priors**; dependency scorer captures **head selection** and **long-range** signals.
- k-best + re-ranking keeps runtime near cubic for the first stage, with small overhead in re-ranking.

Applications

- **IE/QA:** reliable heads (**dependency**) with usable spans (**constituency**) for argument extraction.
- **MT/NLG:** phrase boundaries stabilize reordering; dependencies guide predicate structure.

Q12) Propose an efficiency-focused modification to Inside–Outside for large corpora.

NLP_Assignment No. 2

Proposal: Coarse-to-Fine, Block-Sparse, Neural-Pruned Inside–Outside (CF-BS-NP IO)

Key ingredients

1. **Neural supertagging for span & rule pruning.**
   - A lightweight transformer predicts, for each span `(i,j)`, a top-K set of likely **nonterminals** and **binary rules**. Prune others → **block-sparse charts**.
2. **Coarse-to-fine grammar cascade.**
   - Start with a compact **coarse grammar** to compute cheap inside/outside and **eliminate low-probability spans**; refine only survivors with the **fine grammar**.
3. **Batch-wise GPU dynamic programming.**
   - Represent chart cells as **dense blocks** for surviving labels; compute inside/outside with batched tensor ops (minimizing Python loops).
4. **Expected-count sampling (stochastic EM).**
   - On massive corpora, at each EM iteration sample:
     - A subset of sentences, **and**
     - A subset of spans per sentence (importance-weighted by coarse inside probabilities).
   - Yields an **unbiased estimate** of expected counts while reducing per-iteration cost.
5. **Low-rank parameterization of rule probs.**
   - Factorize `P(A→BC)` via low-rank embeddings of A,B,C to reduce parameter size and speed M-step normalization.
6. **Safe pruning guarantees.**
   - Keep a **fallback** ε-mass for pruned events to preserve likelihood monotonicity; re-introduce pruned items if dev likelihood stagnates ("**elastic pruning**").

Complexity & benefits

- Dramatically fewer span/label combinations touch the chart → **substantially sub-cubic wall-time** in practice.
- **GPU-friendly** batched arithmetic accelerates both E- and M-steps.
- Maintains EM's **monotonic likelihood** increase while scaling to web-scale unlabeled text.

Where it helps

- Training PCFG/constituency components for the **hybrid parser** in Q11 or for semi-supervised grammar induction on **domain-specific corpora** (biomed, legal, code).