NLP – Unit 5 (NLP Tools and Techniques) – END-SEM PYQ Answers

May–June 2023

Q5(a) WordNet [10 Marks]

- **WordNet Definition:** A large-scale, machine-readable lexical database for English organised around synsets (synonym sets). Each synset represents exactly one lexical concept and is linked to related synsets via semantic relations. Developed by George Miller at Princeton University (1985+).

- **Synset (Synonym Set) — Core Unit of WordNet:**
  - A synset = a group of synonymous word forms (lemmas) that all express the same lexical concept
  - Each synset has: (1) list of lemma names, (2) part-of-speech (n/v/adj/adv), (3) gloss (definition), (4) usage examples, (5) semantic relations to other synsets
  - Example: Synset('car.n.01') = {car, auto, automobile, machine, motorcar}
  - Gloss: 'a motor vehicle with four wheels; usually propelled by an internal combustion engine'

- **Semantic Relations in WordNet:**

Relation Name	Direction	Description	Example
Hypernym (IS-A)	Upward (more general)	Parent concept	car IS-A motor vehicle IS-A vehicle IS-A artifact
Hyponym (kind-of)	Downward (more specific)	Child concept — what specific types exist?	sedan, SUV, convertible ARE hyponyms of car
Holonym (part-of)	Whole → Part	The larger whole that contains this entity	car IS HOLONYM OF wheel
Meronym (has-part)	Part → Whole	Parts that make up this entity	wheel, engine, door ARE meronyms of car
Antonym	Opposite	Semantically opposite word	hot ↔ cold, fast ↔ slow, love ↔ hate
Entailment	Implication	Action A necessarily implies action B	snoring ENTAILS sleeping
Also-see	Informal link	Loosely related concepts in same domain	honest also-see trustworthy

- **WordNet IS-A Hierarchy (Noun Ontology):**

```
entity
  object.n.01
    artifact.n.01
      conveyance.n.03
        vehicle.n.01
          motor_vehicle.n.01
            car.n.01  ← target synset
              sedan.n.01
              SUV.n.01
              hatchback.n.01
```

- **Semantic Similarity Metrics Using WordNet:**

Metric	Approach	Best For
Path Length	sim = 1 / (path_length + 1)	Quick baseline
Leacock-Chodorow	sim = −log(path / (2×depth))	Normalises by max depth
Wu-Palmer	sim = 2×depth(LCS) / (depth(s_1) + depth(s_2))	Standard metric
Resnik	sim = IC(LCS)	Corpus-dependent; information content
Jiang-Conrath	sim = 1 / (IC(s_1) + IC(s_2) − 2×IC(LCS))	State-of-art knowledge-based

- **Applications of WordNet:**

Application	How WordNet Helps	Example
WSD	Use synset definitions+relations to find correct sense	'bank' → Lesk algorithm uses WordNet glosses
Query Expansion (IR)	Expand query with synonyms from synset	'car' query also retrieves 'automobile' documents
Semantic Similarity	Path length between synsets = inverse distance	sim(car, motorcycle) > sim(car, mountain)
Ontology for AI	IS-A hierarchy ready for knowledge-based reasoning	'dog IS-A mammal' automatically entails dog properties
Machine Translation	Map source language synsets to target synsets	English 'car.n.01' → French 'voiture.n.01'
Text Classification	Use hypernym features to generalise	'Elon Musk' → PERSON → helps classify tech-person articles

- **WordNet Limitations:**
  - English-only: Princeton WordNet covers only English; cross-lingual use requires multilingual WordNets (EuroWordNet, IndoWordNet)
  - Domain gaps: medical, legal, scientific jargon poorly covered — no ‘myocardial infarction’ with full semantic relations

- Static resource: does not update automatically for neologisms (selfie, tweet-as-verb, COVID-19)
- Formal definitions: glosses are dictionary-style — short and sometimes insufficient for NLP tasks like Lesk

> *WordNet is one of the most cited NLP resources (20,000+ paper citations). The key data structure: synsets connected by semantic relations form a graph (a hierarchy for IS-A, a general graph for other relations). Python access: nltk.corpus.wordnet; wn.synsets('bank') returns all synsets for 'bank'.*

Q5(b) NLP Tools: NLTK and TextBlob Features [7 Marks]

Tool	Primary Focus	Key Strength
NLTK	Research, education, comprehensive	WordNet access, 20+ corpora, many algorithms
spaCy	Production-grade NLP pipeline	Speed, NER, dependency parsing, industrial use
TextBlob	Simple API for beginners	Sentiment, spell check, Google Translate
Gensim	Topic modeling and word vectors	LDA, Word2Vec, doc2vec, scalable

- **NLTK — Detailed Feature Table:**

Feature	Function	Code / Notes
Tokenisation	Split text into word or sentence tokens	word_tokenize(text), sent_tokenize(text)
POS Tagging	Label each word with its part-of-speech	pos_tag(tokens) → [('cat','NN'), ('ran','VBD')]
Stemming	Reduce word to root stem (may not be real word)	PorterStemmer().stem('running') → 'run'
Lemmatisation	Return dictionary base form	WordNetLemmatizer().lemmatize('ran','v') → 'run'
Stopword Removal	Remove high-frequency uninformative words	stopwords.words('english') = ['the','is','a',…]
Chunking	Group tokens into phrase chunks (NP, VP)	RegexpParser with grammar pattern rules
WordNet Interface	Access synsets, hypernyms, hyponyms, similarity	wn.synsets('bank'), wn.path_similarity(s_1, s_2)
Corpus Access	20+ built-in annotated corpora	brown.words(), reuters.fileids(), semcor.sents()
Frequency Distribution	Count word/bigram frequencies	FreqDist(words), ConditionalFreqDist(pairs)

- **TextBlob — Detailed Feature Table:**

Feature	Description	Code Example
Sentiment Analysis	.sentiment → (polarity, subjectivity) tuple	polarity in [−1, +1]; subjectivity in [0, 1]
POS Tagging	.tags → list of (word, POS_tag) tuples	[('Great','JJ'),('movie','NN')]
Noun Phrases	.noun_phrases → key multi-word phrases	['natural language','machine learning']
Spell Check	.correct() → spell-corrected string	'Speling errur'.correct() → 'Spelling error'
Translation	.translate(to='fr') → translated string	Uses Google Translate API
Language Detection	.detect_language() → language code	TextBlob('Bonjour').detect_language() → 'fr'
N-grams	.ngrams(n=2) → list of n-word tuples	Extract bigrams/trigrams for n-gram analysis

Choosing between tools: NLTK for research, education, and WordNet access. TextBlob for simple scripts and prototyping. spaCy for production systems (fast, accurate). Gensim for topic modelling. HuggingFace Transformers for state-of-art results. Many real projects combine 2–3 of these tools.

Q6(a) Lesk Algorithm and Walker's Algorithm for WSD [10 Marks]

- **Word Sense Disambiguation (WSD):** Automatically determine which sense of an ambiguous word is used in a given context.
  - Most Frequent Sense (MFS) baseline: always choose the most common WordNet sense. Achieves ~60–65% accuracy — all WSD systems must beat this.

- **Lesk Algorithm (1986) — Definition:** Select the word sense whose dictionary definition overlaps most with the words in the surrounding context.

- **Lesk Algorithm Steps:**
  - Step 1 – Identify target word (to disambiguate) and context words (surrounding sentence words, minus stopwords)
  - Step 2 – Retrieve all WordNet synsets for the target word: {s_1, s_2, …, s_n}
  - Step 3 – For each synset s_i: extract content words from its gloss (definition)
  - Step 4 – Compute overlap: score(s_i) = |words_in_gloss(s_i) ∩ context_words|
  - Step 5 – Select best sense: s* = argmax$_{s\{sub:i\}}$ score(s_i)
  - Step 6 – If all overlaps = 0: fall back to Most Frequent Sense

- **Lesk Worked Example — Target: 'bank' in 'I deposited my salary at the bank':**

```
Context words (stopwords removed): {deposited, salary, bank}

Sense 1 (Financial): 'a financial institution that accepts deposits and
                      channels the money into lending activities'
  Gloss words: {financial, institution, accepts, deposits, channels, money, lending}
```

```
  Overlap with context: {deposits} ∩ {deposited} ≈ 1  (stem: deposit)

Sense 2 (River): 'sloping land alongside a body of water'
  Gloss words: {sloping, land, alongside, body, water}
  Overlap: {} ∩ {deposited, salary} = 0

Sense 3 (Blood bank): 'a place where blood plasma is stored'
  Gloss words: {place, blood, plasma, stored}
  Overlap: {} ∩ {deposited, salary} = 0

Winner: Sense 1 (Financial Bank) — score = 1 vs 0 vs 0        CORRECT
```

- **Walker's Algorithm — Definition:** Extends Lesk by building an expanded SIGNATURE for each sense (using all related WordNet synsets' glosses) and by considering senses of context words — not just raw context word strings.

- **Walker's Algorithm Steps:**
  - Step 1 – Enumerate all senses of target word: $\{s_1, s_2, \ldots, s_n\}$
  - Step 2 – For each sense s_i: build SIGNATURE = gloss(s_i) + gloss of all related synsets (hypernyms, hyponyms, holonyms, antonyms, siblings)
  - Step 3 – For each context word c_j: identify its best-guess sense (run Lesk on c_j, or use MFS)
  - Step 4 – Score(s_i) = Σ_j overlap(signature(s_i), signature(best_sense(c_j)))
  - Step 5 – $s^* = \text{argmax}_{s\{sub:i\}}$ Score(s_i)

- **Comparison Table — Lesk vs Walker's:**

Aspect	Lesk Algorithm	Walker's Algorithm
Definition source	Direct gloss of target sense only	Gloss + all related synsets' glosses (signature)
Context processing	Raw context words as bag	Context words with their own best senses (recursive)
Overlap computation	target_gloss ∩ context_words	signature(s_i) ∩ signature(sense(c_j)) per context word
Accuracy	30–50% (often below MFS baseline)	35–55% (5–15% improvement over Lesk)
Computational cost	Fast: O(|senses| × |context|)	Slower: signature expansion + per-context sense lookup
Handles synonyms?	No — exact word match only	Better — related synsets contain more vocabulary
Handles polysemous context words?	No	Yes — finds best sense of context words too

Both Lesk and Walker's are KNOWLEDGE-BASED (unsupervised) WSD methods — they require only WordNet, no labelled training data. Supervised WSD (LSTM, BERT fine-tuned on SemCor) achieves 75–80%

accuracy. The SemCor corpus contains 234K words tagged with WordNet senses — the primary training resource for supervised WSD.

Q6(b) Gensim Library: Tasks and Examples [7 Marks]

- **Gensim:** Open-source Python library for unsupervised topic modelling and NLP, specialising in scalable algorithms for large text corpora. Built for memory-efficient corpus streaming.

Task Category	Algorithm	Gensim Tool	Notes
Word Embeddings	Word2Vec (CBOW / Skip-gram)	gensim.models.Word2Vec	Learns word vectors from local context windows
Word Embeddings	FastText (sub-word + word)	gensim.models.FastText	Handles OOV via character n-grams
Doc Embeddings	Doc2Vec (PV-DM / PV-DBOW)	gensim.models.Doc2Vec	Trains document-level embeddings
Topic Modelling	LDA (Latent Dirichlet Allocation)	gensim.models.LdaModel	Probabilistic topic discovery
Topic Modelling	LSI/LSA	gensim.models.LsiModel	Algebraic SVD-based topic modelling
Topic Modelling	NMF	gensim.models.Nmf	Faster topic modelling for short texts
Phrase Detection	Bigram/Trigram collocations	gensim.models.Phrases	Detects 'New_York', 'machine_learning'
Similarity Search	Sparse/dense similarity queries	gensim.similarities	Find most similar documents to a query

- **Gensim Word2Vec Example:**

```
from gensim.models import Word2Vec

sentences = [
  ['nlp', 'is', 'fascinating'],
  ['deep', 'learning', 'transforms', 'nlp'],
]

# Train Skip-gram (sg=1): 100-dim vectors, window=5
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, sg=1, epochs=10)

model.wv.most_similar('nlp', topn=3)
# [('deep', 0.95), ('learning', 0.92), ('embeddings', 0.88)]

# Analogy: king − man + woman ≈ queen
result = model.wv.most_similar(positive=['king','woman'], negative=['man'])
print(result[0])  # ('queen', 0.89)
```

- **Gensim LDA Example:**

```
from gensim import corpora, models

docs = [
  ['nlp', 'language', 'model'],
  ['neural', 'deep', 'learning'],
]
dictionary = corpora.Dictionary(docs)
bow_corpus = [dictionary.doc2bow(doc) for doc in docs]

lda = models.LdaModel(bow_corpus, num_topics=2,
                      id2word=dictionary, passes=20)
for idx, topic in lda.print_topics(num_words=5):
    print(f'Topic {idx}: {topic}')
```

Gensim's LdaModel supports online (mini-batch) training and can stream corpora from disk without loading everything into memory — handles corpora too large for RAM (Wikipedia, Common Crawl). For production topic modelling at web scale, Gensim is the standard Python library.

November–December 2023

Q5(a) NLP Tools: Features of Any 3 Tools [7 Marks]

- **Three Selected Tools: NLTK, spaCy, Gensim**

- **spaCy — Feature Table:**

Feature	spaCy Details
Tokeniser	Rule + ML pipeline: handles contractions, URLs, emails; 65+ language tokenisers
NER	Pre-trained neural NER (en_core_web_sm/md/lg): 18 entity types, ~90% F1 on OntoNotes
POS Tagging	Fine-grained Penn Treebank tags (NN, VBZ) + Universal POS (NOUN, VERB) simultaneously
Dependency Parsing	Arc-eager neural dependency parser; labeled relations (nsubj, obj, det, amod)
Word Vectors	Built-in GloVe vectors in md/lg models; token.vector attribute
Speed	Very fast: 100K+ words/second; production-ready

REPEATED — Refer to: May–June 2023 → Q5(b) [NLTK 9-feature table with code examples]

REPEATED — Refer to: May–June 2023 → Q6(b) [Gensim task table + Word2Vec + LDA code examples]

Q5(b) Lesk and Walker's Algorithm for WSD [10 Marks]

REPEATED — Refer to: May–June 2023 → Q6(a) [Complete Lesk + Walker's with worked example + comparison table]

Q6(a) Lexical Knowledge Networks [10 Marks]

- **Lexical Knowledge Networks:** Structured databases representing word meanings, properties, and semantic relationships in machine-readable form.

- **i) WordNet:**

REPEATED — Refer to: May–June 2023 → Q5(a) [Full WordNet — synsets, 7 relation types, hierarchy, applications, limitations]

- **ii) IndoWordNet (IIT Bombay):**
- **Definition:** Multilingual lexical knowledge base covering 18 scheduled Indian languages, all linked through a shared synset ID system derived from Princeton WordNet.

Aspect	Princeton WordNet	IndoWordNet
Languages covered	English only	18+ Indian languages (Hindi, Marathi, Tamil, Telugu, Bengali…)
Synset count	~117,000 English synsets	Hindi ~50K, Marathi ~35K, Tamil ~30K (varies)
Construction	Original (built from scratch by linguists)	Expansion (translate EN synsets to Indian lang + verify)
Scripts	Latin (Roman) only	Devanagari, Tamil, Telugu, Kannada scripts…
Cross-lingual links	No (monolingual)	Yes — all Indian language synsets share Princeton WN synset IDs
Morphological detail	Limited (English is not highly inflected)	Rich: especially for agglutinative languages (Tamil, Kannada)

- **Unique Challenges for IndoWordNet:**
  - Script diversity: 13 different scripts for 18 languages — requires careful Unicode handling
  - Morphological complexity: agglutinative languages (Tamil, Kannada, Telugu) have thousands of inflected forms per root
  - Diglossic variation: formal literary language differs significantly from colloquial spoken forms (Tamil diglossia is extreme)
  - Limited digital resources: some languages (Bodo, Santali, Manipuri) lack sufficient digitised text for validation
  - Translation ambiguity: English synsets don't always have direct equivalents in Indian languages (cultural concepts)

- **iii) VerbNet:**
- **Definition:** A hierarchical class-based verb lexicon that organises English verbs into classes based on shared syntactic frames and semantic predicates.

```
VerbNet Class: give-13.1
Members: give, hand, pass, deliver, forward, transfer

Frame 1: NP V NP to NP
 'John gave the book to Mary'
 Agent=John, Theme=book, Recipient=Mary
 Semantics: cause(Agent, transfer(Agent,Theme,Recipient))

Frame 2: NP V NP NP
 'John gave Mary the book'  [same semantics, different surface form]
```

- **iv) PropBank:**
- **Definition:** Penn Treebank sentences annotated with semantic role labels for verbal predicates — who did what to whom, when, where, how.

```
'Google acquired YouTube in 2006 for $1.65 billion'
Predicate: acquired (acquire.01)
 Arg0: Google          = Agent (acquirer)
 Arg1: YouTube         = Theme (entity acquired)
 ArgM-TMP: in 2006     = Temporal modifier
 ArgM-MNR: for $1.65B  = Manner/Price modifier
```

- **v) Treebanks:**
  - Penn Treebank (PTB): 4.5M words from WSJ (1989–1990); phrase-structure (constituency) trees; 48 POS tags
  - Universal Dependencies (UD): 250+ treebanks in 100+ languages; consistent dependency annotation; 37 dependency relation types
  - UD evaluation: UAS (Unlabelled Attachment Score) = % tokens with correct head; LAS = % with correct head AND label

> *These five resources form the backbone of knowledge-based NLP: WordNet for lexical semantics, IndoWordNet for multilingual coverage, VerbNet for verbal semantics, PropBank for semantic role annotations, Treebanks for syntactic structure. Together they represent 40+ person-years of expert linguistic annotation.*

Q6(b) NLTK Tokenisation Code [7 Marks]

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize, WordPunctTokenizer

text = "I can't stop learning NLP! It's amazing, isn't it?"

# METHOD 1: Whitespace — split ONLY on whitespace; punctuation stays attached
whitespace_tokens = text.split()
# ["I", "can't", "stop", "learning", "NLP!", "It's", "amazing,", "isn't", "it?"]

# METHOD 2: WordPunctTokenizer — splits on ALL punctuation
```

```
wpt_tokens = WordPunctTokenizer().tokenize(text)
# ["I", "can", "'", "t", "stop", "NLP", "!", "It", "'", "s", "amazing", ",", ...]

# METHOD 3: word_tokenize — Penn Treebank Tokenizer (RECOMMENDED)
nltk_tokens = word_tokenize(text)
# ["I", "ca", "n't", "stop", "learning", "NLP", "!", "It", "'s", "amazing", ",", "is", "n't", "it", "?"]
```

Criterion	Method 1 (split)	Method 2 (WordPunct)	Method 3 (word_tokenize)
Contraction handling	Kept together (can't)	Split on ' (can, ', t)	Smart split (ca, n't)
Punctuation	Attached to word (NLP!)	Separated as own tokens	Separated intelligently
Speed	Fastest (built-in Python)	Fast	Medium (needs punkt model)
Accuracy	Low (punctuation noise)	Medium (loses contractions)	High (Penn Treebank standard)
Best for	Quick counting, clean text	Need clean alphabetic tokens	Standard NLP preprocessing

May–June 2024

Q5(a) Walker's Algorithm vs Lesk Algorithm [9 Marks]

REPEATED — Refer to: May–June 2023 → Q6(a) [Full Lesk + Walker's with worked example and 9-row comparison table]

Scenario	Why Walker's Helps	Example
Short context (2–3 words)	Direct Lesk finds 0 overlap; Walker's expanded signature has more vocab	'He went to the bank'
Domain-specific text	Hypernym chain adds broader vocabulary matching domain terms	Medical 'bank' of patients
Polysemous context words	Walker's considers senses of context words too	'charged' near 'bank' helps disambiguate financial sense
WordNet-rich nouns	Nouns have deep hierarchies → large signatures	'hospital', 'disease' have many hypernyms with relevant glosses

Q5(b) IndoWordNet vs Traditional WordNet [8 Marks]

REPEATED — Refer to: Nov–Dec 2023 → Q6(a) ii [IndoWordNet vs WordNet 6-column comparison table + challenges]

- **Additional — IndoWordNet Architecture:**
  - Shared synset ID system: every IndoWordNet synset has the same integer ID as the corresponding Princeton WordNet synset

- This enables cross-lingual NLP: given a word in Hindi, find its WN synset ID → look up English words → find Tamil words for same ID
- Each language adds: lemma names in that language's script, gloss, usage example, pronunciation, and morphological information

Q6(a) NLTK vs spaCy vs TextBlob Comparison [9 Marks]

Feature	NLTK	spaCy	TextBlob
Primary Purpose	Research, education, comprehensive toolkit	Production NLP pipeline (fast + accurate)	Simple beginner-friendly API
Speed	Slow (pure Python)	Very fast (C/Cython backend)	Medium (wraps NLTK)
Tokenisation	word_tokenize (Penn Treebank)	Rule+ML pipeline, handles edge cases well	Based on NLTK tokeniser internally
POS Tagging	pos_tag (averaged perceptron tagger)	Neural CNN tagger — fine-grained + universal	Uses NLTK's tagger
NER	Basic chunking (low accuracy ~60%)	Pre-trained (18 types, ~90%+ F1)	Limited
Dependency Parsing	Limited (basic chunking only)	Full labeled dependency tree (arc-eager neural)	Not supported
Word Vectors	Not built-in (use Gensim)	Built-in GloVe in md/lg models	Not supported
Sentiment Analysis	VADER (separate) or custom	Not built-in	Built-in: polarity + subjectivity
WordNet Access	Full access: wn.synsets(), path_similarity()	Not direct (use NLTK for WN)	Limited
Corpus Access	20+ built-in corpora (Brown, Reuters, SemCor)	No built-in corpora	No built-in corpora
Best Use Case	Research, WN tasks, algorithm study	Production NLP, fast inference, NER+parsing	Prototyping, sentiment, spell-check

Recommendation by use case: (1) Learning NLP algorithms: NLTK. (2) Building production NLP service: spaCy. (3) Quick text analysis or prototyping: TextBlob. (4) Topic modelling or word embeddings: Gensim. (5) State-of-art performance: HuggingFace Transformers.

Q6(b) PropBank and VerbNet: Significance [8 Marks]

- **PropBank Significance:**
  - Primary training resource for Semantic Role Labelling (SRL): who did what to whom, when, where
  - Enables information extraction: 'Google acquired YouTube' → SRL gives Arg0=Google, Arg1=YouTube
  - Used in QA: 'Who founded Tesla?' → SRL finds the Arg0 of 'found' in sentences about Tesla

- **PropBank Examples:**

```
'Microsoft acquired Activision in 2023'
  Arg0: Microsoft   (Acquirer)
  Arg1: Activision  (Entity acquired)
  ArgM-TMP: in 2023 (Time)

'The CEO resigned from the company yesterday'
  Arg0: The CEO        (Person resigning)
  Arg2: from the company (Employer)
  ArgM-TMP: yesterday    (Time)
```

- **VerbNet Significance:**
  - Verb class generalisation: learning the semantics of 'give' automatically applies to 'hand', 'pass', 'deliver' (all in give-13.1)
  - Syntax–semantics interface: VerbNet maps surface sentence patterns to semantic predicates
  - Feature generation: VerbNet class membership is a useful feature for SRL, IE, MT reranking

> *PropBank + VerbNet together form the basis of SRL benchmarks. State-of-art SRL (BERT + biaffine attention) achieves 85%+ F1 on CoNLL-2005 and CoNLL-2012 benchmarks.*

May–June 2025 [6404]-96

Q5(a) NLTK, spaCy, Gensim: Tokenisation and POS Tagging [9 Marks]

REPEATED — Refer to: May–June 2024 and earlier sessions — see all comparison tables above

Use Case	Recommended Tool	Reason
Learning NLP from scratch	NLTK	Educational, explicit algorithms, WordNet access
Fast production NER + parsing	spaCy	Speed, accuracy, industrial-grade models
Sentiment analysis prototype	TextBlob	1-line API; polarity + subjectivity out-of-box
Topic modelling on 1M+ docs	Gensim	Streaming corpus, LDA, word vectors, scalable
BERT fine-tuning / inference	HuggingFace	All transformer models, AutoModel, Trainer API

Q5(b) WordNet as Lexical Knowledge Network [8 Marks]

REPEATED — Refer to: May–June 2023 → Q5(a) [Complete WordNet — synsets, 7 relations, hierarchy, applications, limitations]

Q6(a) Treebanks and Universal Dependency Treebanks [9 Marks]

- **Penn Treebank (PTB):** 4.5M words from WSJ (1989–1990); phrase-structure (constituency) trees; 48 POS tags; gold standard for English parsing research.
- **Universal Dependencies (UD):** 250+ treebanks in 100+ languages; consistent dependency annotation; 37 dependency relation types; enables cross-lingual NLP research.
- **UD CoNLL-U Format:**

```
# Fields: ID  FORM  LEMMA  UPOS  XPOS  FEATS  HEAD  DEPREL  DEPS  MISC
# Sentence: 'The cat sat'
1  The  the  DET   DT   Definite=Def  2  det   _ _
2  cat  cat  NOUN  NN   Number=Sing   3  nsubj  _ _
3  sat  sit  VERB  VBD  Tense=Past    0  root   _ _

HEAD=0 means root of the sentence
LAS (Labelled Attachment Score) = % tokens with correct HEAD + DEPREL
```

Q6(b) Lesk Algorithm for WSD [8 Marks]

REPEATED — Refer to: May–June 2023 → Q6(a) [Lesk full explanation with worked example and comparison table]

November–December 2025

Q5(a) Walker's Algorithm for WSD [9 Marks]

REPEATED — Refer to: May–June 2023 → Q6(a) [Walker's full explanation with steps and comparison to Lesk]

- **Additional — Walker's Handles Polysemous Context Words:**

```
Sentence: 'I went to the bank near the river'
Target: 'bank' | Context: {went, near, river}

Walker's Step 3: Get best sense of EACH context word:
 'river' Sense 1 (watercourse): 'a large natural stream of water'
   Signature: {water, stream, flow, natural, banks, current, …}

Walker's Step 4: Score overlap for each bank sense:
 bank.sense1 (financial) signature: {money, deposits, finance, institution, …}
   Overlap with river-sense signature: minimal
 bank.sense2 (river-bank) signature: {water, sloping, land, stream, riverbank…}
   Overlap with river-sense signature: {water, stream} → score = 2!

Winner: Sense 2 (River Bank)      correctly disambiguated by 'river' context
```

Q5(b) NLP Libraries: NLTK and Gensim [9 Marks]

REPEATED — Refer to: May–June 2023 → Q5(b) [NLTK features] + Q6(b) [Gensim tasks and code]

Task	NLTK	Gensim
Tokenisation	word_tokenize() — full-featured	tokenize() — simple alphabetic
POS Tagging	pos_tag() — averaged perceptron	NOT SUPPORTED natively
NER	ne_chunk() — basic, limited	NOT SUPPORTED
WordNet Access	Full access via corpus.wordnet	NOT SUPPORTED
Word Embeddings	Not built-in	Word2Vec, FastText, GloVe loading
Topic Modelling	Not built-in	LDA, LSI, NMF — primary strength
Corpus Scale	Manageable (fits in RAM)	Millions of docs (streaming corpus)

Q6(a) IndoWordNet vs Traditional WordNet [9 Marks]

REPEATED — Refer to: Nov–Dec 2023 → Q6(a) ii + May–June 2024 → Q5(b) [Full IndoWordNet comparison + challenges + architecture]

Q6(b) Gensim Library Tasks with Examples [9 Marks]

REPEATED — Refer to: May–June 2023 → Q6(b) [Full Gensim task table + Word2Vec code + LDA code]

Topic Frequency Analysis — Unit 5

Rank	Topic	Sessions	Priority
1	Lesk + Walker's WSD Algorithm	MJ23, ND23, MJ24, MJ25, ND25	VERY HIGH — 5× every exam
2	NLP Libraries (NLTK/spaCy/TextBlob/Gensim)	MJ23, ND23, MJ24, MJ25, ND25	VERY HIGH — 5× every exam
3	WordNet (synsets, relations, hierarchy)	MJ23, ND23, MJ24, MJ25	HIGH — 4×
4	IndoWordNet vs Traditional WordNet	ND23, MJ24, ND25	HIGH — 3×
5	PropBank + VerbNet (semantic roles)	ND23, MJ24	MEDIUM — 2×
6	Treebanks + Universal Dependencies	ND23, MJ25	MEDIUM — 2×
7	NLTK Tokenisation Code	ND23	LOW — 1×

EXAM STRATEGY — Unit 5: (1) Lesk: always write the worked example with 'bank' in a financial context — show 3 senses with gloss words and overlap computation. (2) Walker's: explain the signature expansion concept and why it beats Lesk in short contexts. (3) WordNet: memorise all 7 relation names (hypernym, hyponym, holonym, meronym, antonym, entailment, also-see) with examples. (4) NLTK vs spaCy vs TextBlob: spaCy for speed/production, NLTK for research/WordNet, TextBlob for sentiment/beginners. (5) IndoWordNet: know the 3 main challenges (script diversity, morphological complexity, diglossic variation).