NLP Unit-4

Description Logics (DL)

Definition:

DL is used for structuring and reasoning about knowledge, commonly in AI and semantic web applications.

Real-Life Example:

A medical ontology can classify diseases based on symptoms and treatments.

A smart home system can infer that if a room is occupied and the temperature is low, the heater should turn on.

Demo program

```
from owlready2 import *


onto = get_ontology("http://example.org/ontology.owl")


with onto:
    class Disease(Thing): pass
    class Symptom(Thing): pass
    class hasSymptom(Disease >> Symptom): pass
```

```
covid = Disease("COVID-19")

fever = Symptom("Fever")

covid.hasSymptom.append(fever)


print(list(covid.hasSymptom))  # Output: [Fever]
```

Description Logics

Description Logics (DL) is a formalism used to represent structured knowledge and reason about it. It is commonly used in ontologies, such as OWL (Web Ontology Language).

Example (Using OWL and RDF

```
from owlready2 import *

# Create ontology
onto = get_ontology("http://example.org/ontology.owl")

with onto:
    class Person(Thing): pass
    class hasChild(Person >> Person): pass

# Create individuals
john = Person("John")
jane = Person("Jane")
john.hasChild.append(jane)

onto.save(file="ontology.owl", format="rdfxml")
print(list(john.hasChild))  # Output: [Jane]
```

Syntax-Driven Semantic Analysis

This technique assigns semantic structure based on syntactic structures, often using **context-free grammars (CFGs) and parsing**.

```
import nltk
from nltk import CFG

grammar = CFG.fromstring("""
    S -> NP VP
    NP -> 'Alice' | 'Bob'
    VP -> 'runs' | 'jumps'
""")

parser = nltk.ChartParser(grammar)

sentence = ['Alice', 'runs']
for tree in parser.parse(sentence):
    print(tree)
```

Syntax-Driven Semantic Analysis

Definition:

Assigns meaning to sentences based on their structure.

Real-Life Example:

"John saw Mary with a telescope."

Does John have a telescope, or does Mary?

Syntax analysis helps disambiguate this.

Syntax-Driven Semantic Analysis

```
import nltk
from nltk import CFG

grammar = CFG.fromstring("""
    S -> NP VP
    NP -> 'John' | 'Mary' | 'telescope'
    VP -> 'saw' NP
""")

parser = nltk.ChartParser(grammar)

sentence = ['John', 'saw', 'Mary']
for tree in parser.parse(sentence):
    print(tree)
```

Semantic Attachments

Definition:

Attaching meaning or functions to grammar rules.

Real-Life Example:

If a sentence contains "buy", it means a financial transaction.

In a chatbot, recognizing "weather" should trigger a weather API call.

Semantic Attachments

```
from nltk.sem import Expression

read_expr = Expression.fromstring


expr = read_expr('buy(John, Laptop)')

print(expr)  # Meaning: John buys a Laptop
```

3. Semantic Attachments

Semantic attachments add **meaning to syntactic structures**, often using functions or rules.

```
from nltk.sem import Expression

read_expr = Expression.fromstring


expr1 = read_expr('runs(Alice)')

expr2 = read_expr('jumps(Bob)')


print(expr1, expr2)
```

4. Word Senses

Definition:

Words have multiple meanings (polysemy).

Real-Life Example:

"Bank" can mean a financial institution or a riverbank.

"Apple" can mean a fruit or a tech company.

4. Word Senses

Words can have multiple senses, which leads to ambiguity.

https://wordnet.princeton.edu/

https://www.naukri.com/code360/library/wordnet-in-nlp

https://medium.com/@akankshamalhotra24/a-short-note-on-wordnet-aa97a7194ecf

```
from nltk.corpus import wordnet


synsets = wordnet.synsets("bank")
for syn in synsets:
    print(syn, syn.definition())
```

WordNet

Category	Unique Forms	Number of Senses
Noun	94474	116317
Verb	10319	22066
Adjective	20170	29881
Adverb	4546	5677

Semantic Analysis

The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.

We already know that lexical analysis also deals with the meaning of the words, then how is semantic analysis different from lexical analysis? Lexical analysis is based on smaller token but on the other side semantic analysis focuses on larger chunks. That is why semantic analysis can be divided into the following two parts

Semantic Analysis

Studying meaning of individual word

It is the first part of the semantic analysis in which the study of the meaning of individual words is performed. This part is called lexical semantics.

Studying the combination of individual words

In the second part, the individual words will be combined to provide meaning in sentences.

The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyze the sentence Ram is great. In this sentence, the speaker is talking either about Lord Ram or about a person whose name is Ram. That is why the job, to get the proper meaning of the sentence, of semantic analyzer is important.

Relations Between Senses

Definition:

Words relate to each other through synonyms, antonyms, hypernyms, and hyponyms.

Real-Life Example:

Synonymy: "Big" and "Large"

Antonymy: "Hot" and "Cold"

Hypernymy (Generalization): "Dog" → "Animal"

Hyponymy (Specialization): "Rose" → "Flower"

5. Relations Between Senses

- Synonymy: Similar words
- Antonymy: Opposite words
- Hypernymy: Generalization
- Hyponymy: Specialization

```
synset = wordnet.synsets("car")[0]
print("Synonyms:", synset.lemma_names())
print("Hypernyms:", synset.hypernyms())
print("Hyponyms:", synset.hyponyms())
```

The noun "bass" has 8 senses in WordNet.

1. bass - (the lowest part of the musical range)
2. bass, bass part - (the lowest part in polyphonic music)
3. bass, basso - (an adult male singer with the lowest voice)
4. sea bass, bass - (flesh of lean-fleshed saltwater fish of the family Serranidae)
5. freshwater bass, bass - (any of various North American lean-fleshed freshwater fishes especially of the genus Micropterus)
6. bass, bass voice, basso - (the lowest adult male singing voice)
7. bass - (the member with the lowest range of a family of musical instruments)
8. bass - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Thematic Roles

Definition:

Assigns roles to words in a sentence:

- Agent (doer): "John" in "John eats an apple."
- Theme (object affected): "Apple"
- Recipient (receiver): "Mary" in "John gave Mary a book."

Real-Life Example:

In Google Voice Assistant, it identifies the agent, action, and object when you say, "Play music on Spotify."

6. Thematic Roles

Thematic roles assign semantic roles to sentence elements (e.g., Agent, Theme).

```
import spacy
nlp = spacy.load("en_core_web_sm")


sentence = nlp("John gave Mary a book.")
for token in sentence:
    print(token.text, token.dep_)
```

7. Selectional Restrictions

Definition:
Rules that determine which words can logically go together.

Real-Life Example:
"He drank water" ✅

"He drank a rock" ❌ (Incorrect because drinking requires a liquid)

These restrict what arguments a verb can take.

```
synset = wordnet.synsets("eat")[0]

print(synset.verb_groups())  # Shows related verbs
```

Word Sense Disambiguation (WSD)

Definition:

Determining the correct sense of a word in context.

Real-Life Example:

"He went to the bank to deposit money." (Bank = Financial)

"He sat by the bank of the river." (Bank = Riverside)

Word Sense Disambiguation

We understand that words have different meanings based on the context of its usage in the sentence. If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence.

Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of word is activated by the use of word in a particular context. Lexical ambiguity, syntactic or semantic, is one of the very first problem that any NLP system faces.

Part-of-speech (POS) taggers with high level of accuracy can solve Words syntactic ambiguity. On the other hand, the problem of resolving semantic ambiguity is called WSD (word sense disambiguation).

Resolving semantic ambiguity is harder than resolving syntactic ambiguity.

For example

For example, consider the two examples of the distinct sense that exist for the word bass −

I can hear bass sound.

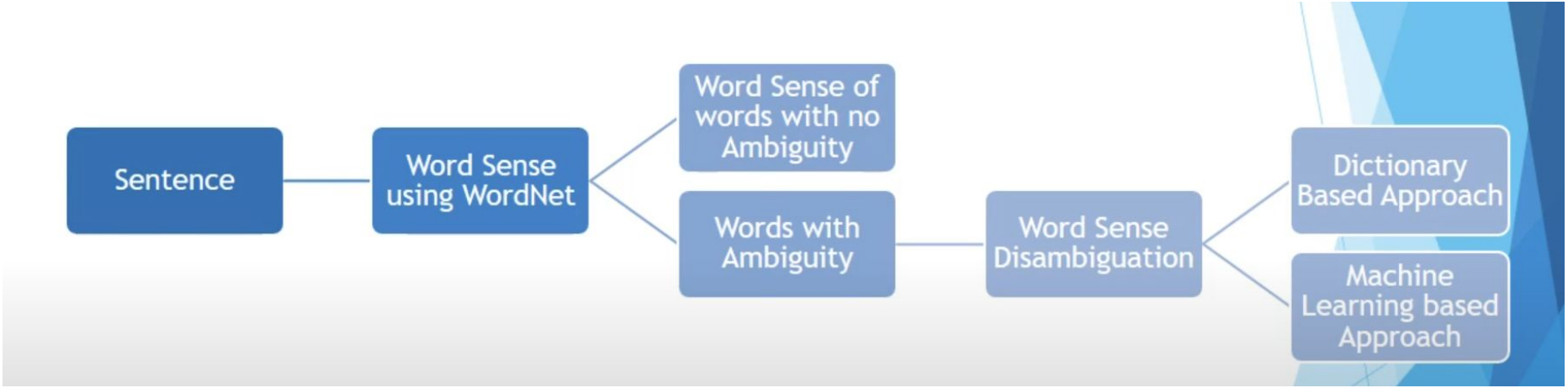
He likes to eat grilled bass.

The occurrence of the word bass clearly denotes the distinct meaning. In first sentence, it means frequency and in second, it means fish. Hence, if it would be disambiguated by WSD then the correct meaning to the above sentences can be assigned as follows −

I can hear bass/frequency sound.

He likes to eat grilled bass/fish.

Word Sense Disambiguation



Evaluation of WSD

The evaluation of WSD requires the following two inputs −

A Dictionary

The very first input for evaluation of WSD is dictionary, which is used to specify the senses to be disambiguated.

Test Corpus

Another input required by WSD is the high-annotated test corpus that has the target or correct-senses. The test corpora can be of two types & minsu;

Lexical sample − This kind of corpora is used in the system, where it is required to disambiguate a small sample of words.

All-words − This kind of corpora is used in the system, where it is expected to disambiguate all the words in a piece of running text.

Approaches and Methods to Word Sense Disambiguation (WSD)

Dictionary-based or Knowledge-based Methods

As the name suggests, for disambiguation, these methods primarily rely on dictionaries, treasures and lexical knowledge base. They do not use corpora evidences for disambiguation.

The Lesk method is the seminal dictionary-based method introduced by Michael Lesk in 1986.

The Lesk definition, on which the Lesk algorithm is based is measure overlap between sense definitions for all words in context.

However, in 2000, Kilgarriff and Rosensweig gave the simplified Lesk definition as measure overlap between sense definitions of word and current context, which further means identify the correct sense for one word at a time.

Here the current context is the set of words in surrounding sentence or paragraph.

Supervised Methods

For disambiguation, machine learning methods make use of sense-annotated corpora to train. These methods assume that the context can provide enough evidence on its own to disambiguate the sense. In these methods, the words knowledge and reasoning are deemed unnecessary. The context is represented as a set of features of the words. It includes the information about the surrounding words also. Support vector machine and memory-based learning are the most successful supervised learning approaches to WSD. These methods rely on substantial amount of manually sense-tagged corpora, which is very expensive to create.

Semi-supervised Methods

Due to the lack of training corpus, most of the word sense disambiguation algorithms use semi-supervised learning methods. It is because semi-supervised methods use both labelled as well as unlabeled data. These methods require very small amount of annotated text and large amount of plain unannotated text. The technique that is used by semisupervised methods is bootstrapping from seed data.

Unsupervised Methods

These methods assume that similar senses occur in similar context. That is why the senses can be induced from text by clustering word occurrences by using some measure of similarity of the context.

This task is called word sense induction or discrimination. Unsupervised methods have great potential to overcome the knowledge acquisition bottleneck due to non-dependency on manual efforts.

Applications of Word Sense Disambiguation (WSD)

Machine Translation

Machine translation or MT is the most obvious application of WSD. In MT, Lexical choice for the words that have distinct translations for different senses, is done by WSD. The senses in MT are represented as words in the target language. Most of the machine translation systems do not use explicit WSD module.

Information Retrieval (IR)

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information. The system basically assists users in finding the information they required but it does not explicitly return the answers of the questions. WSD is used to resolve the ambiguities of the queries provided to IR system. As like MT, current IR systems do not explicitly use WSD module and they rely on the concept that user would type enough context in the query to only retrieve relevant documents.

Text Mining and Information Extraction (IE)

In most of the applications, WSD is necessary to do accurate analysis of text. For example, WSD helps intelligent gathering system to do flagging of the correct words. For example, medical intelligent system might need flagging of illegal drugs rather than medical drugs

Lexicography

WSD and lexicography can work together in loop because modern lexicography is corpusbased. With lexicography, WSD provides rough empirical sense groupings as well as statistically significant contextual indicators of sense.

Difficulties in Word Sense Disambiguation (WSD)

Differences between dictionaries

The major problem of WSD is to decide the sense of the word because different senses can be very closely related. Even different dictionaries and thesauruses can provide different divisions of words into senses.

Different algorithms for different applications

Another problem of WSD is that completely different algorithm might be needed for different applications. For example, in machine translation, it takes the form of target word selection; and in information retrieval, a sense inventory is not required.

Inter-judge variance

Another problem of WSD is that WSD systems are generally tested by having their results on a task compared against the task of human beings. This is called the problem of interjudge variance.

Word-sense discreteness

Another difficulty in WSD is that words cannot be easily divided into discrete submeanings.

Lesk Algorithm

Lesk Algorithm is a classical Word Sense Disambiguation algorithm introduced by Michael E. Lesk in 1986.

The Lesk algorithm is based on the idea that words in a given region of the text will have a similar meaning. In the Simplified Lesk Algorithm, the correct meaning of each word context is found by getting the sense which overlaps the most among the given context and its dictionary meaning.

8. Word Sense Disambiguation (WSD)

Supervised WSD

Train a classifier on labeled data.

```
from nltk.wsd import lesk

sentence = "He went to the bank to deposit money".split()

sense = lesk(sentence, "bank")

print(sense, sense.definition())
```

```
c1= lesk(word_tokenize('Water current'),'current')
print(c1,c1.definition())
```

```
c1= lesk(word_tokenize('The current time is 2 AM'),'current')
print(c1,c1.definition())
```

Dictionary & Thesaurus-Based WSD

Uses lexical resources like WordNet.

Real-Life Example:

A dictionary lookup for a chatbot to select word meanings.

9. Dictionary and Thesaurus-Based WSD

Uses lexical resources like WordNet and thesauruses.

print(wordnet.synsets("bank")) #

Dictionary in NLP

A dictionary in NLP is a collection of words (or tokens) mapped to some kind of linguistic or semantic information.

- Types of NLP Dictionaries:

Lexical Dictionaries – Provide word meanings, POS tags, lemmas, etc.

Example: WordNet, Merriam-Webster API, Oxford Dictionary API

Morphological Dictionaries – Contain root words and their inflected forms.

Used in stemming and lemmatization.

Domain-specific Dictionaries – Contain terms relevant to a particular domain (e.g., medical, legal).

Custom Dictionaries – Created for tasks like Named Entity Recognition (NER), sentiment classification, etc.

Uses in NLP:

- ❏ Token normalization (lemmatization/stemming)
- ❏ Part-of-Speech tagging
- ❏ Named Entity Recognition
- ❏ Word sense disambiguation
- ❏ Spell-check and autocorrect

Thesaurus in NLP

A **thesaurus** is a resource that lists **synonyms, antonyms, and related words** for each word.

🔹 **Common NLP Thesauri:**

- **WordNet**: The most widely used electronic lexical database with synonym sets (synsets).
- **Roget's Thesaurus**: Classical resource, less structured for computational use.
- **ConceptNet**: Semantic network that relates words and concepts.

✅ **Uses in NLP:**

- Query expansion in search engines
- Paraphrase generation
- Text summarization
- Semantic similarity and clustering
- Word sense disambiguation
- Chatbot/Conversational AI vocabulary enhancement

syn,defi,example..

```
from nltk.corpus import wordnet as wn

# Get synonyms for "bright"

synonyms = set()

for syn in wn.synsets("bright"):

    for lemma in syn.lemmas():

        synonyms.add(lemma.name())

print("Synonyms of 'bright':", synonyms)

# Get definition and example

syn = wn.synsets("bright")[0]

print("Definition:", syn.definition())

print("Example:", syn.examples())
```

```
import nltk
from nltk.corpus import wordnet as wn

# Make sure WordNet is downloaded
nltk.download('wordnet')

# Get synonyms for "bright"
synonyms = set()
for syn in wn.synsets("bright"):
    for lemma in syn.lemmas():
        synonyms.add(lemma.name())

print(synonyms)
```

Word Similarity using Thesaurus and Distributional methods.

Difference between Thesaurus Based & Distributional

Feature	Thesaurus-Based	Distributional
Data dependency	No (manual)	Yes (needs large corpora)
Interpretability	High	Lower
Contextual awareness	Low	High
Speed	Fast (if preloaded)	Moderate
Flexibility	Limited to known words	Generalizable

Example

```
import nltk
nltk.download('punkt_tab')
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
a1= lesk(word_tokenize('This device is used to jam the signal'),'jam')
print(a1,a1.definition())
a2 = lesk(word_tokenize('I am stuck in a traffic jam'),'jam')
print(a2,a2.definition())
```

Example

```
# testing with some data
b1= lesk(word_tokenize('Apply spices to the chicken to season it'),'season')
print(b1,b1.definition())


b2= lesk(word_tokenize('India receives a lot of rain in the rainy season'),'season')
print(b2,b2.definition())
```

These rely on structured linguistic resources like WordNet or ConceptNet.

✅ Characteristics:

Use predefined relationships like synonymy, hypernymy, antonymy, etc.

Work well even with small corpora.

Not data-driven—don't capture context-based meaning variation.

🔧 **Common Thesaurus-Based Similarity Measures:**

Path Similarity – Shortest path between words in the WordNet graph.

Wu-Palmer Similarity – Based on depth of the least common ancestor.

Leacock-Chodorow Similarity – Based on shortest path and depth of taxonomy.

Word Similarity via path & Wup

```
from nltk.corpus import wordnet as wn

word1 = wn.synsets('car')[0]

word2 = wn.synsets('automobile')[0]


# Path Similarity

print("Path Similarity:", word1.path_similarity(word2))

# Wu-Palmer Similarity

print("Wu-Palmer Similarity:", word1.wup_similarity(word2))
```

```
Path Similarity: 1.0
Wu-Palmer Similarity: 1.0
```

Path Similarity

Concept:

Measures similarity based on the shortest path between two concepts (synsets) in the WordNet taxonomy graph.

similarity = 1 / (path_length + 1)

Working:

- Each synset is a node.
- The edges represent "is-a" (hypernym/hyponym) relationships.
- If the synsets are the same, path length is 0 → similarity = 1.
- The more nodes between them, the lower the similarity.

```
from nltk.corpus import wordnet as wn
dog = wn.synset('dog.n.01')
cat = wn.synset('cat.n.01')
print(dog.path_similarity(cat))  # Output: e.g., 0.2
```

Wu-Palmer Similarity

Concept:

Measures similarity based on the depth of the least common ancestor (LCA) in the taxonomy and the depth of the individual synsets.

similarity = 2 * depth(LCS) / (depth(synset1) + depth(synset2))

Working:

Deeper LCA → More specific shared meaning → Higher similarity.

Values range from 0 (no similarity) to 1 (identical synsets).

```
print(dog.wup_similarity(cat))
```

Leacock-Chodorow (LCH) Similarity

Concept:

Uses the shortest path length and the maximum depth of the taxonomy to calculate similarity.

similarity = -log(path_length / (2 * max_depth))

Working:

- Requires synsets to be from the same part of speech (e.g., both nouns).
- Uses logarithmic scale for similarity.
- Higher depth and shorter path → Higher similarity.

```
print(dog.lch_similarity(cat))
```

Path Similarity / Wui-Palmer / Leacock-Chodorow

Similarity Measure	Based On	Value Range	Notes
Path Similarity	Shortest path in hierarchy	0 to 1	Sensitive to taxonomic distance
Wu-Palmer	Depth of LCA and synsets	0 to 1	High if both words share a specific ancestor
Leacock-Chodorow	Shortest path + taxonomy depth	≥ 0 (log scale)	Uses log to scale similarity

Distributional Methods

These are based on the **distributional hypothesis**: *"Words that occur in similar contexts tend to have similar meanings."*

✅ Characteristics:

- Data-driven
- Captures **contextual** and **semantic** similarity
- Needs large corpora
- Common in modern NLP (especially with word embeddings)

🔧 Common Techniques:

1. **Count-Based Vectors** (TF-IDF, co-occurrence matrices)
2. **Word Embeddings** (Word2Vec, GloVe, FastText)
3. **Contextual Embeddings** (BERT, RoBERTa, etc.)

KeyedVectors

```
from gensim.models import KeyedVectors

# Load pretrained word2vec model (GoogleNews-vectors-negative300.bin.gz)

# This file is large (~1.5GB) and must be downloaded separately

model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary=True)

similarity = model.similarity('king', 'queen')

print("Similarity between 'king' and 'queen':", similarity)

# Most similar words to "virus"

print(model.most_similar('virus', topn=5))
```

```
import gensim.downloader as api


model = api.load("glove-wiki-gigaword-100")
similarity = model.similarity("teacher", "instructor")
print("GloVe similarity:", similarity)
```