**EXPERIMENT - 6**

**Aim**:

1. Find synonyms and hyponyms using Python nltk and WordNet

2. Get all the hyponyms and hypernyms for a given word

3. Get all Hyponyms with synsetID

4. Implement Word sense disambiguation using Sysnet

**Theory**:

Wordnet is a large collection of words and vocabulary from the English language that are related to each other and are grouped in some way. That’s the reason WordNet is also called a lexical database. WordNet groups nouns, adjectives, verbs which are similar and calls them synsets or synonyms. A group of synsets might belong to some other synset. Synset is a special kind of a simple interface that is present in NLTK to look up words in WordNet. Synset instances are the groupings of synonymous words that express the same concept. For example, the synsets “Brick” and “concrete” belong to the synset “Construction Materials” or the synset “Brick” also belongs to another synset called “brickwork “. In the example given, brick and concrete are called hyponyms of synset construction materials and also the synsets construction material and brickwork are called synonyms.

Word Sense Disambiguation is an important method of NLP by which the meaning of a word is determined, which is used in a particular context. NLP systems often face the challenge of properly identifying words, and determining the specific usage of a word in a particular sentence has many applications.

Word Sense Disambiguation basically solves the ambiguity that arises in determining the meaning of the same word used in different situations.

**How to implement WSD?**

There are four main ways to implement WSD.

· **Dictionary- and knowledge-based methods**

o These methods rely on text data like dictionaries, thesaurus, etc. It is based on the fact that words that are related to each other can be found in the definitions. The popularly used Lesk method, which is a seminal dictionary-based method.

· **Supervised methods**

o In this type, sense-annotated corpora are used to train machine learning models. But a problem that may arise is that such corpora are very tough and time-consuming to create.

· **Semi-supervised Methods**

o Due to the lack of such corpus, most word sense disambiguation algorithms use semi-supervised methods. The process starts with a small amount of data, which is often manually created.

o This is used to train an initial classifier. This classifier is used on an untagged part of the corpus, to create a larger training set. Basically, this method involves bootstrapping from the initial data, which is referred to as the seed data.

o Semi-supervised methods thus, use both labeled and unlabelled data.

· **Unsupervised Methods**

o Unsupervised Methods pose the greatest challenge to researchers and NLP professionals. A key assumption of these models is that similar meanings and senses occur in a similar context. They are not dependent on manual efforts, hence can overcome the knowledge acquisition deadlock.

**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.

**Code:**

| import nltk  from nltk.corpus import wordnet as wn    nltk.download("omw-1.4")  nltk.download("wordnet")    synonyms = []  word = "call"  for syn in wn.synsets(word):  for i in syn.lemmas():  synonyms.append(i.name())  print(f"Synonyms of {word} are: ")  print(set(synonyms), "\n")    hyponyms = []  print(f"Hyponyms of {word} are: ")  for syn in wn.synsets(word):  for i in syn.hyponyms():  hyponyms.append(i.name().split(".")[0])  print(set(hyponyms), "\n")    hypernyms = []  for syn in wn.synsets(word):  # print(syn.hypernym\_distances())  for i in syn.hypernym\_distances():  hypernyms.append((i[0].name().split(".")[0], i[1]))  print(f"Hypernyms of {word} are: ")  print(set(hypernyms), "\n")    vehicle = wn.synset("call.n.01")  typesOfVehicles = list(set([w for s in vehicle.closure(lambda s: s.hyponyms()) for w in s.lemma\_names()]))  print(f"Hyponyms of Synset ID {vehicle} are: ")  print(set(typesOfVehicles)) |
| --- |

| import nltk  nltk.download("omw-1.4")  nltk.download("wordnet")  from nltk.corpus import wordnet as wn  from nltk.wsd import lesk  sentence = ["I", "went", "to", "the", "track", "to", "watch", "the", "car", "race", "."]  print(lesk(sentence, "track", "n"), "\n")  print(lesk(sentence, "track"), "\n")  syns = wn.synsets("track")  print(syns, "\n")  # print the word  print(syns[0].lemmas()[0].name(), "\n")  from nltk.corpus import wordnet as wn  for ss in wn.synsets("track"):  print(ss, ss.definition())  print("\n")  print([(s, s.pos()) for s in wn.synsets("can")], "\n")  sentence = "I can kill seventy monsters in a minute".split()  print(lesk(sentence, "can"), "\n")  print(lesk(sentence, "can", pos="v"), "\n")  print(lesk("Ram adores Site".split(), "adores", synsets=[])) |
| --- |

**Output:**

| Synonyms of call are:  {'cry', 'margin\_call', 'scream', 'telephone', 'bid', 'prognosticate', 'send\_for', 'call\_option', 'phone', 'name', 'song', 'yell', 'ring', 'telephone\_call', 'visit', 'anticipate', 'birdcall', 'call\_off', 'phone\_call', 'hollo', 'squall', 'address', 'holler', 'vociferation', 'claim', 'Call', 'call\_up', 'call', 'shout\_out', 'foretell', 'shout', 'outcry', 'promise', 'call\_in', 'predict', 'birdsong', 'forebode'}  Hyponyms of call are:  {'howl', 'crank\_call', 'second-guess', 'call-back', 'read', 'scream', 'beep', 'conference\_call', 'term', 'bell-like\_call', 'hoot', 'entitle', 'summon', 'yelling', 'preempt', 'yodel', 'style', 'baptize', 'prophesy', 'double', 'forecast', 'clamor', 'long\_distance', 'shriek', 'catcall', 'hollo', 'war\_cry', 'misname', 'underbid', 'hail', 'call-in', 'rename', 'raise', 'lift', 'local\_call', 'function\_call', 'cell\_phone', 'outbid', 'call\_up', 'tag', 'boo', 'squawk', 'hurrah', 'overbid', 'wake-up\_call', 'halloo', 'dub', 'muster', 'augur', 'round', 'outcall', 'system\_call', 'two-note\_call', 'collect\_call', 'blue\_murder', 'recall', 'call\_in', 'bellow', 'hosanna', 'whoop', 'refer', 'drop\_by', 'noise', 'bet', 'see', 'post'}  Hypernyms of call are:  {('shout', 0), ('message', 3), ('attitude', 2), ('writing', 5), ('communicate', 4), ('order', 1), ('communication', 3), ('transfer', 4), ('visit', 1), ('call\_option', 0), ('act', 3), ('call', 0), ('ask', 3), ('break', 3), ('interact', 4), ('entice', 1), ('transfer', 6), ('request', 1), ('address', 0), ('act', 5), ('medium', 3), ('instrumentality', 4), ('assembly', 3), ('utter', 1), ('think', 2), ('postpone', 1), ('communicate', 8), ('inform', 2), ('entity', 5), ('communication', 7), ('abstraction', 4), ('request', 3), ('name', 0), ('coding\_system', 3), ('convey', 9), ('psychological\_feature', 7), ('think', 4), ('move', 11), ('interact', 8), ('event', 4), ('transfer', 10), ('abstraction', 6), ('evaluate', 2), ('legal\_document', 3), ('declare', 1), ('request', 5), ('talk', 2), ('option', 1), ('writing', 4), ('challenge', 1), ('artifact', 5), ('communicate', 3), ('code', 2), ('event', 6), ('decision', 1), ('speech\_act', 2), ('abstraction', 8), ('designate', 2), ('request', 7), ('derivative\_instrument', 2), ('convey', 4), ('challenge', 3), ('meeting', 2), ('move', 6), ('invite', 4), ('ask', 2), ('communicate', 5), ('written\_communication', 6), ('demand', 2), ('transfer', 5), ('birdcall', 0), ('predict', 0), ('talk', 6), ('provoke', 2), ('convey', 6), ('indicate', 1), ('group\_action', 4), ('psychological\_feature', 4), ('move', 8), ('communicate', 7), ('wait', 3), ('request', 2), ('whole', 6), ('margin\_call', 0), ('entity', 7), ('cry', 0), ('physical\_entity', 8), ('speculate', 2), ('signal', 2), ('change', 5), ('psychological\_feature', 6), ('think', 3), ('stop', 1), ('ask', 6), ('act', 9), ('choice', 2), ('cognition', 3), ('request', 4), ('label', 1), ('entity', 9), ('visit', 0), ('end', 4), ('oppose', 3), ('document', 4), ('reason', 3), ('telephone', 1), ('event', 5), ('interact', 3), ('evaluate', 3), ('communication', 4), ('telecommunication', 2), ('object', 7), ('convey', 3), ('see', 1), ('inclination', 1), ('move', 5), ('ask', 1), ('act', 4), ('written\_communication', 5), ('demand', 1), ('telecommunicate', 1), ('instruction', 1), ('contest', 2), ('communication', 6), ('transfer', 7), ('address', 1), ('convey', 5), ('change', 2), ('move', 7), ('delay', 2), ('meet', 1), ('animal\_communication', 1), ('entity', 6), ('claim', 2), ('awaken', 1), ('abstraction', 5), ('play', 1), ('auditory\_communication', 2), ('argue', 4), ('converse', 5), ('bid', 0), ('psychological\_feature', 5), ('utterance', 1), ('interrupt', 2), ('communicate', 2), ('read', 1), ('entity', 8), ('compete', 2), ('guess', 1), ('abstraction', 7), ('action', 3)}  Hyponyms of Synset ID Synset('call.n.01') are:  {'call-in', 'crank\_call', 'call-back', 'trunk\_call', 'conference\_call', 'collect\_call', 'local\_call', 'toll\_call', 'long\_distance', 'three-way\_calling', 'wake-up\_call', 'long-distance\_call'} |
| --- |

| [nltk\_data] Downloading package omw-1.4 to /root/nltk\_data...  [nltk\_data] Downloading package wordnet to /root/nltk\_data...  Synset('track.n.11')  Synset('track.n.11')  [Synset('path.n.04'), Synset('lead.n.03'), Synset('track.n.03'), Synset('racetrack.n.01'), Synset('cut.n.08'), Synset('track.n.06'), Synset('track.n.07'), Synset('track.n.08'), Synset('track.n.09'), Synset('track.n.10'), Synset('track.n.11'), Synset('track.v.01'), Synset('track.v.02'), Synset('chase.v.01'), Synset('traverse.v.01'), Synset('track.v.05')]  path  Synset('path.n.04') a line or route along which something travels or moves  Synset('lead.n.03') evidence pointing to a possible solution  Synset('track.n.03') a pair of parallel rails providing a runway for wheels  Synset('racetrack.n.01') a course over which races are run  Synset('cut.n.08') a distinct selection of music from a recording or a compact disc  Synset('track.n.06') an endless metal belt on which tracked vehicles move over the ground  Synset('track.n.07') (computer science) one of the circular magnetic paths on a magnetic disk that serve as a guide for writing and reading data  Synset('track.n.08') a groove on a phonograph recording  Synset('track.n.09') a bar or pair of parallel bars of rolled steel making the railway along which railroad cars or other vehicles can roll  Synset('track.n.10') any road or path affording passage especially a rough one  Synset('track.n.11') the act of participating in an athletic competition involving running on a track  Synset('track.v.01') carry on the feet and deposit  Synset('track.v.02') observe or plot the moving path of something  Synset('chase.v.01') go after with the intent to catch  Synset('traverse.v.01') travel across or pass over  Synset('track.v.05') make tracks upon  [(Synset('can.n.01'), 'n'), (Synset('can.n.02'), 'n'), (Synset('can.n.03'), 'n'), (Synset('buttocks.n.01'), 'n'), (Synset('toilet.n.02'), 'n'), (Synset('toilet.n.01'), 'n'), (Synset('can.v.01'), 'v'), (Synset('displace.v.03'), 'v')]  Synset('can.v.01')  Synset('can.v.01')  None |
| --- |

**Conclusion**:Hence, word sense disambiguation has been performed using WordNet, sysnet and NLTK.