M04S01 Introduction to Semantic Processing

NLP – Semantic Processing

1. Introduction
2. Concepts and Terms
3. Entity and Entity Types
4. Arity and Reification
5. Schema
6. Semantic Associations
7. Databases - WordNet and ConceptNet
8. Word Sense Disambiguation - Naive Bayes
9. Word Sense Disambiguation - Lesk Algorithm
10. Lesk Algorithm Implementation
11. Summary
12. Graded Questions

Introduction

Previous two modules focused on lexical and syntactic processing techniques

Lexical Processing

- Regular expressions
- Tokenization, Stemming, Lemmatization
- TF-IDF model
- Phonetic hashing
- The minimum edit distance algorithm

Syntactic Processing

- POS tagging and HMMs
- CFGs and PCFGs
- Dependency Parsing
- Information Extraction (NER using CRFs and other techniques)

Semantic Processing cover techniques and algorithms to infer the meaning of a given piece of text.

Probably the most challenging area in the field of NLP, partly because the concept of 'meaning' itself is quite wide, and because it is a genuinely hard problem to make machines understand text the same way as we humans do - inferring the intent of a statement, meanings of ambiguous words, dealing with synonyms, detecting sarcasm and so on.

Techniques to solve some common semantic processing problems in NLP.

In this Module

- **Introduction to semantic text processing**: Defining meaning, understanding concepts, terms, entities, relations between entities etc.
- **Vector Semantics**: Representing words as vectors
- **Topic Modelling**: Identifying 'topics' being talked about in text

In this Session

- What is semantics and Why it is important
- Semantic associations
- Word sense disambiguation
- The Lesk algorithm

Concepts and Terms

Semantic processing is about understanding the meaning of a given piece of text. But what do we mean by 'understanding the meaning' of text? Let's see how the human brain processes meaning.

Sentence:

"Croatia fought hard before succumbing to France's deadly attack; lost the finals 2 goals to 4",

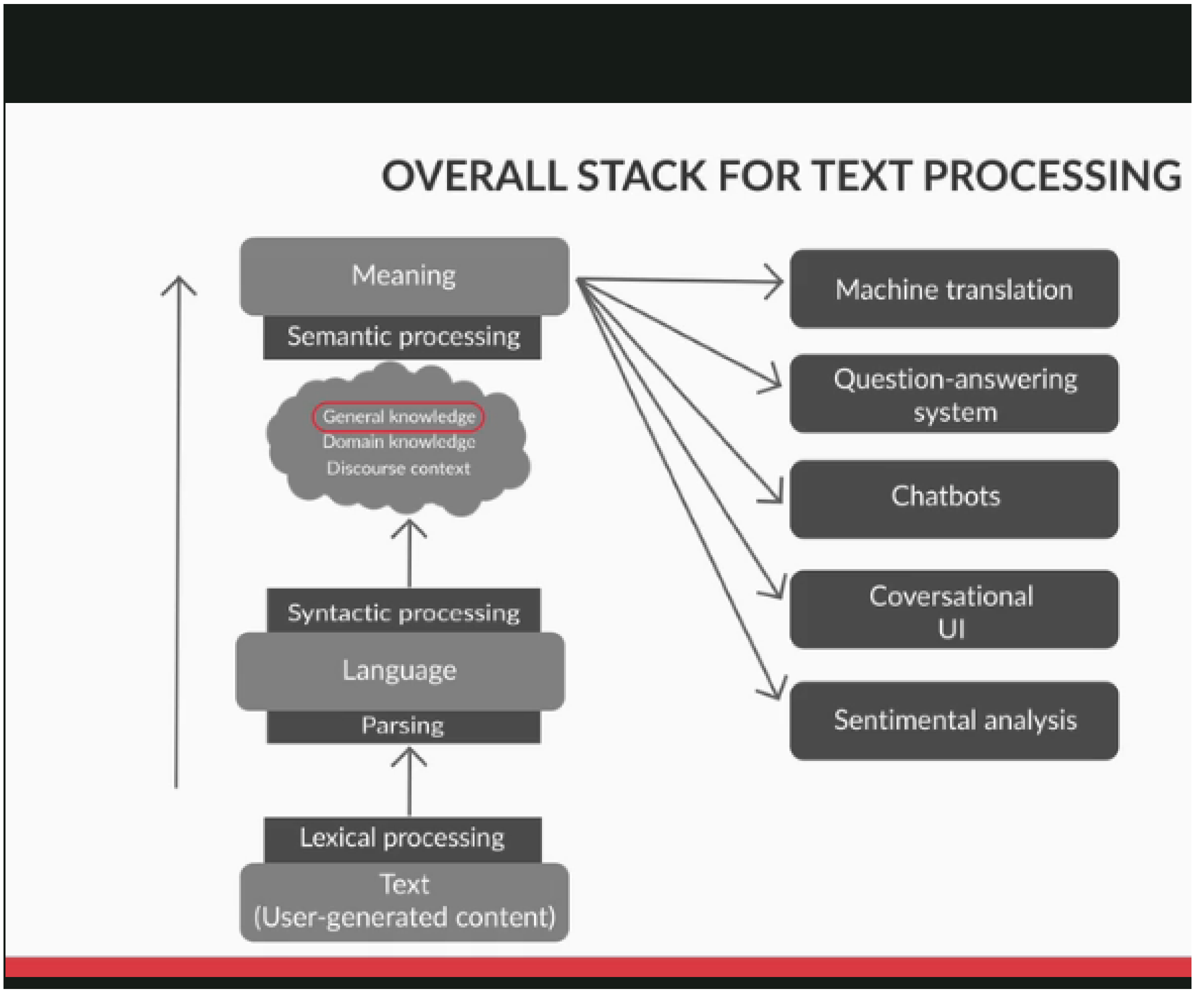
Understood that it's about football and the FIFA world cup final, even though the words 'football' and 'FIFA' are not mentioned. Also, words 'succumb' and 'goal' are used differently than in the sentences '*He succumbed to head injuries and died on the spot*' and '*My life goals*'.

Brain can process sentences meaningfully because it can relate the text to other words and concepts it already knows, such as football, FIFA etc. It can process meaning in the **context of the text** and can **disambiguate between multiple possible senses** of words such as 'goal' and 'succumb'. Also, brain understands **topics being talked about** in a text, such as 'football' and 'FIFA World Cup', even though these exact words are not present in the text.

Semantic text processing focusses on teaching machines to process text in similar ways.

Specifically, some areas of semantics in this course are **word sense disambiguation** (identifying the intended meaning of an ambiguous word), representing **words as vectors** semantically like other words, **topic modelling** (identifying topics being talked about in documents) etc.

But before all that, build a basic understanding of the question - "what is meaning?".



OVERALL STACK FOR TEXT PROCESSING

Meaning

Semantic processing

General knowledge
Domain knowledge
Discourse context

Syntactic processing

Language

Parsing

Lexical processing

Text
(User-generated content)

Machine translation

Question-answering system

Chatbots

Coversational UI

Sentimental analysis

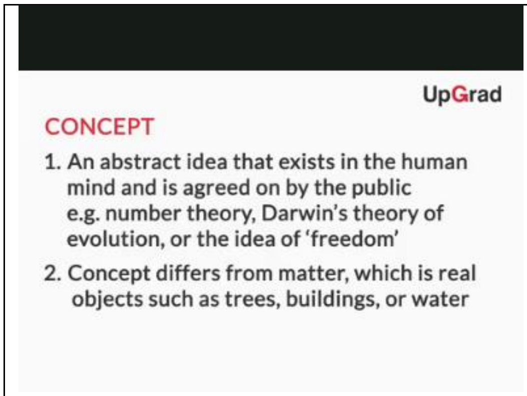
To study semantics, we first need to establish a representation of ‘**meaning’**. Though we often use the term ‘meaning’ quite casually, it is quite non-trivial to answer the question “What is the meaning of meaning, and how do you represent the meaning of a statement?”.

Clearly, we cannot build 'parse trees for meaning' or assign 'meaning tags' to each word. Thus, the first step in semantic processing is to create a model to interpret the 'meaning' of text. This task is non-trivial.

There are objects which exist, but we cannot touch, see or hear them, such as independence, freedom, algebra and so on. But they still do exist and occur in natural language. We refer to these objects as **'concepts'**.

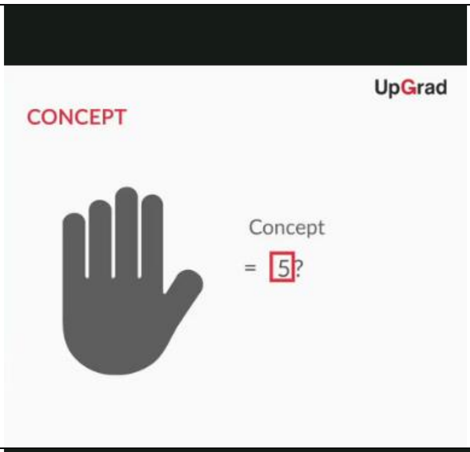
The idea of **'concepts'** forms an important component in the representation of meaning.

Let’s now understand how **concepts** and **terms** are used to define meaning.

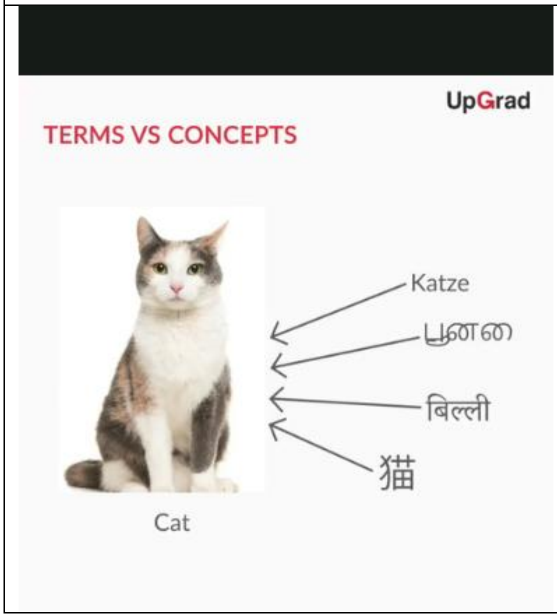


CONCEPT

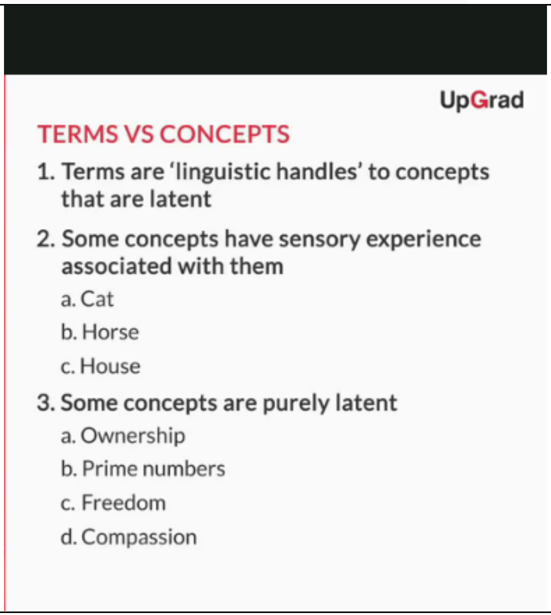
1. An abstract idea that exists in the human mind and is agreed on by the public e.g. number theory, Darwin's theory of evolution, or the idea of 'freedom'
2. Concept differs from matter, which is real objects such as trees, buildings, or water



CONCEPT



TERMS VS CONCEPTS



TERMS VS CONCEPTS

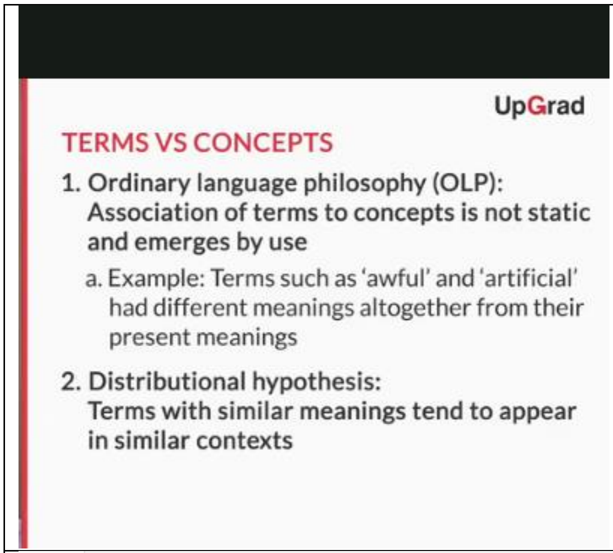
1. Terms are 'linguistic handles' to concepts that are latent
2. Some concepts have sensory experience associated with them
   a. Cat
   b. Horse
   c. House
3. Some concepts are purely latent
   a. Ownership
   b. Prime numbers
   c. Freedom
   d. Compassion

Terms act as *handles* to concepts and the notion of '**concepts'** gives us a way to represent the '**meaning'** of a given text.

But how do terms acquire certain concepts? It turns out that the context in which terms are frequently used make the term acquire a certain meaning. For e.g. the word 'bank' has different meanings in the phrases 'bank of a river' and 'a commercial bank' because the word happens to be used differently in these contexts.

Next, two important concepts to help understand the anatomy of 'meaning' better:

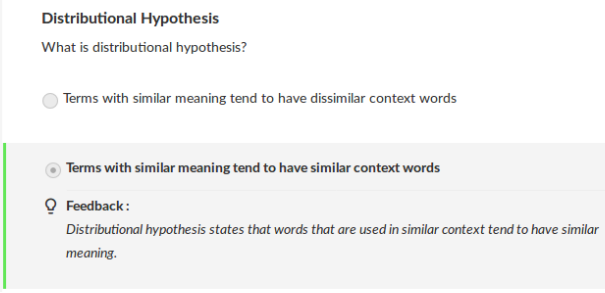
1. Terms which appear in similar contexts are like each other
2. Terms acquire meaning through use in certain contexts, and the meaning of terms may change depending on the context in which they appear



UpGrad

TERMS VS CONCEPTS

1. Ordinary language philosophy (OLP): Association of terms to concepts is not static and emerges by use
   a. Example: Terms such as 'awful' and 'artificial' had different meanings altogether from their present meanings
2. Distributional hypothesis: Terms with similar meanings tend to appear in similar contexts



Distributional Hypothesis

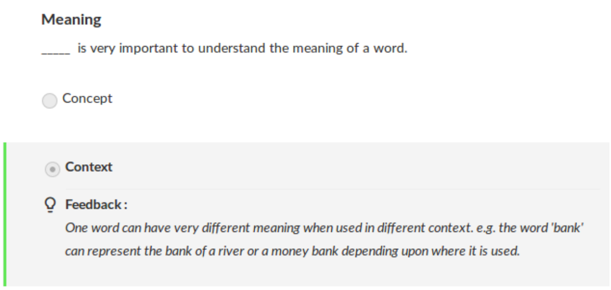
What is distributional hypothesis?

○ Terms with similar meaning tend to have dissimilar context words

● **Terms with similar meaning tend to have similar context words**

Feedback :

Distributional hypothesis states that words that are used in similar context tend to have similar meaning.



Meaning

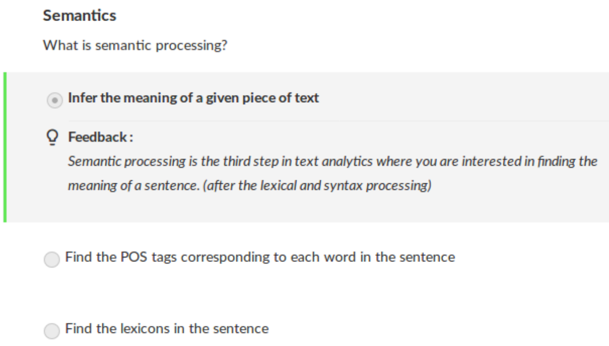
_____ is very important to understand the meaning of a word.

○ Concept

● **Context**

Feedback :

One word can have very different meaning when used in different context. e.g. the word 'bank' can represent the bank of a river or a money bank depending upon where it is used.



Semantics

What is semantic processing?

● **Infer the meaning of a given piece of text**

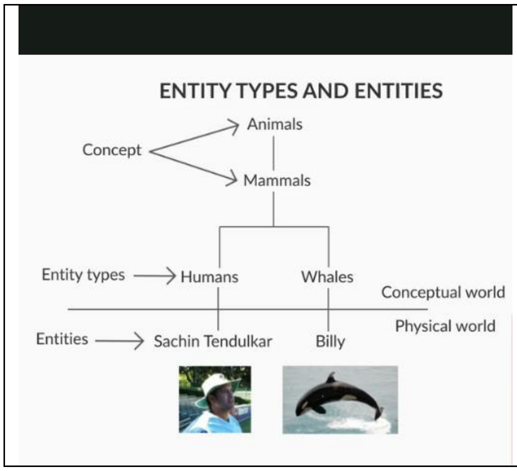
Feedback :

Semantic processing is the third step in text analytics where you are interested in finding the meaning of a sentence. (after the lexical and syntax processing)

○ Find the POS tags corresponding to each word in the sentence

○ Find the lexicons in the sentence

Entity and Entity Types

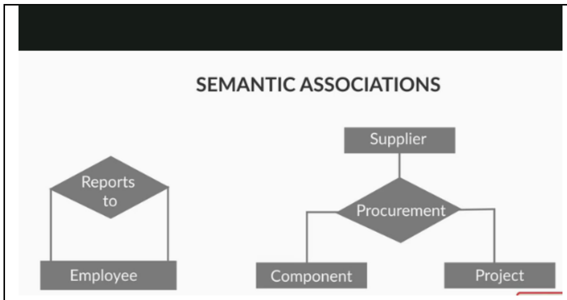
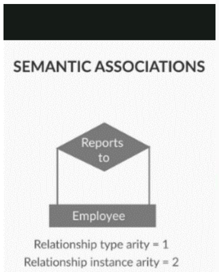
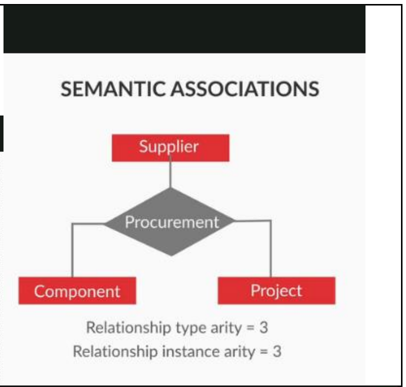
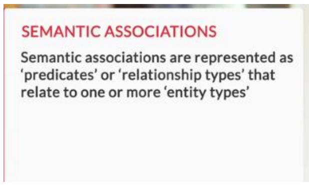
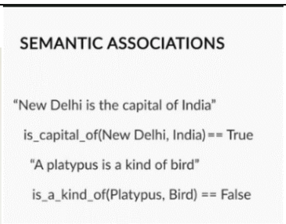
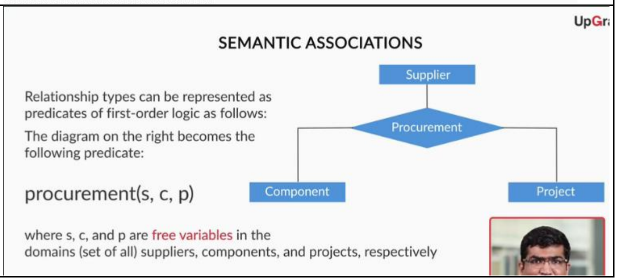
 ENTITY TYPES AND ENTITIES Concept Animals Mammals Entity types Humans Whales Conceptual world Physical world Entities Sachin Tendulkar Billy	**Concepts** and **terms** are closely related to each other - terms acting as handles for concepts. **Entities** and **Entity types** are two other concepts which helps refine representation of meaning further. **Entities** are instances of **entity types**. Multiple entity types can be grouped under a **concept.** This hierarchical model of concepts, entities and entity types can answer real-world questions and process the 'meaning' of sentences.

Associations Between Entity Types

Say you ask Alexa, '*Alexa, what is a Labrador?*', it answers *'It is a breed of dogs* '. Say you ask, *'Who is the coach of the Indian cricket team*?', it reveals the name of the coach.

To answer such questions, a system needs mapping **between entities and entity types**, i.e. it needs to understand that a Labrador is a dog, a mammal is an animal, a coach is a specific person etc.

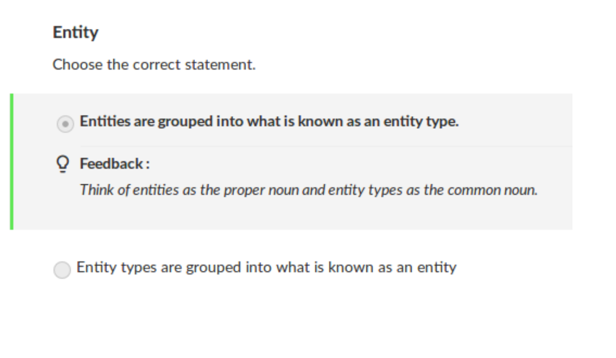
This is solved by **associations** between entities and entity types, represented using the notion of **predicates**.

 SEMANTIC ASSOCIATIONS Reports to Employee Supplier Procurement Component Project	 SEMANTIC ASSOCIATIONS Relationship type arity = 1 Relationship instance arity = 2  SEMANTIC ASSOCIATIONS Relationship type arity = 3 Relationship instance arity = 3
 SEMANTIC ASSOCIATIONS Semantic associations are represented as 'predicates' or 'relationship types' that relate to one or more 'entity types'  SEMANTIC ASSOCIATIONS "New Delhi is the capital of India" is_capital_of(New Delhi, India) == True "A platypus is a kind of bird" is_a_kind_of(Platypus, Bird) == False	 SEMANTIC ASSOCIATIONS Relationship types can be represented as predicates of first-order logic as follows: The diagram on the right becomes the following predicate: procurement(s, c, p) where s, c, and p are free variables in the domains (set of all) suppliers, components, and projects, respectively

The notion of a **predicate** is a simple model to process the meaning of complex statements. For example, say you ask an artificial NLP system - "Did France win the football world cup final in 2018?". The statement can be broken down into a set of predicates, each returning True or False, such as win (France, final) = True, final (FIFA, 2018) = True.

A **predicate** is a function which takes in some parameters and returns True or False depending on the relationship between the parameters. For example, a predicate *teacher_teaches_course(P = professor Srinath, C = text analytics)* returns True.

Processing meaning involves understanding the **associations between entities**. For e.g. "A football team is comprised of 11 players" is an example of relation type =1 (since there is only one entity type - football players), though the relationship instance is 11 (since there are 11 instances of the entity).



Entity

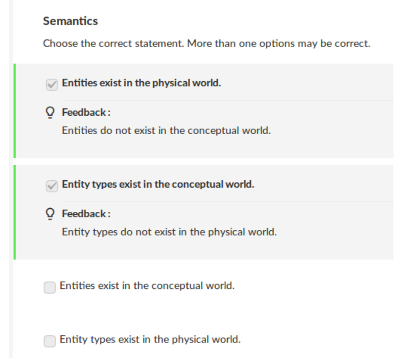
Choose the correct statement.

● **Entities are grouped into what is known as an entity type.**

Feedback :

Think of entities as the proper noun and entity types as the common noun.

○ Entity types are grouped into what is known as an entity



Semantics

Choose the correct statement. More than one options may be correct.

☑ **Entities exist in the physical world.**

Feedback :

Entities do not exist in the conceptual world.

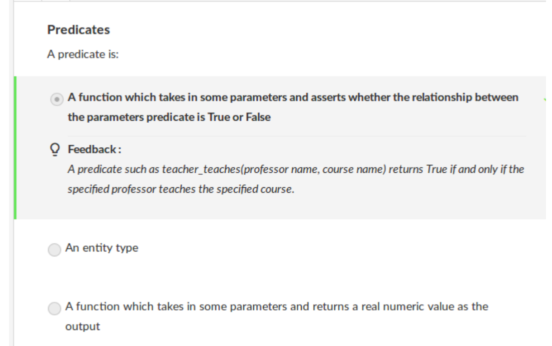
☑ **Entity types exist in the conceptual world.**

Feedback :

Entity types do not exist in the physical world.

□ Entities exist in the conceptual world.

□ Entity types exist in the physical world.



Predicates

A predicate is:

● **A function which takes in some parameters and asserts whether the relationship between the parameters predicate is True or False**

Feedback :

A predicate such as teacher_teaches(professor name, course name) returns True if and only if the specified professor teaches the specified course.

○ An entity type

○ A function which takes in some parameters and returns a real numeric value as the output

Arity and Reification

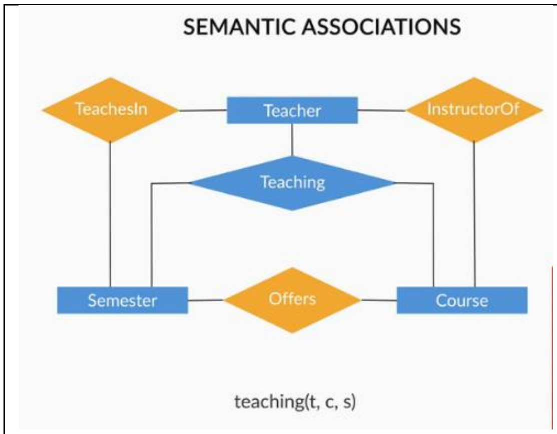
Consider that these three statements are true:

- Shyam supplies cotton to Vivek
- Vivek manufactures t-shirts
- Shyam supplies cotton which is used to manufacture t-shirts

Can we conclude that the following statement is also true - "Shyam supplies cotton to Vivek which he uses to manufacture t-shirts"?

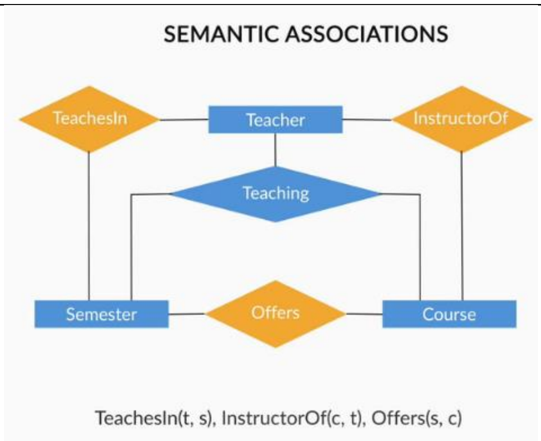
Predicates are assertions that take in some parameters, such as *supplier_manufacturer(Shyam, Vivek)*, and return True or False. But most real-world phenomena are much more complex to be represented by simple **binary predicates**, and so we need to use **higher-order** predicates (such as the **ternary predicate** supplier_manufacturer_product(Shyam, Vivek, t-shirts)).

Further, if a binary predicate is true, it is not necessary that a higher order predicate will also be true. This is captured by the notion of **arity of a predicate**.



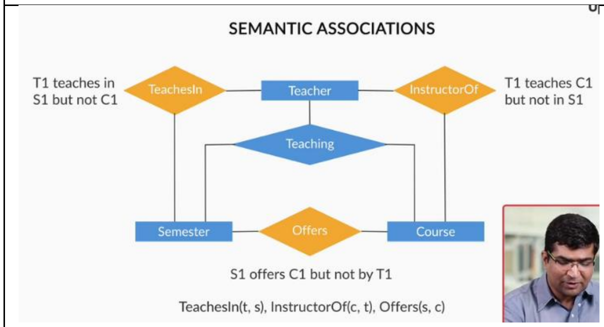
SEMANTIC ASSOCIATIONS

teaching(t, c, s)



SEMANTIC ASSOCIATIONS

TeachesIn(t, s), InstructorOf(c, t), Offers(s, c)



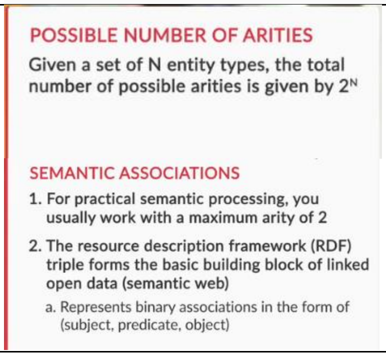
SEMANTIC ASSOCIATIONS

T1 teaches in S1 but not C1

T1 teaches C1 but not in S1

S1 offers C1 but not by T1

TeachesIn(t, s), InstructorOf(c, t), Offers(s, c)



POSSIBLE NUMBER OF ARITIES

Given a set of N entity types, the total number of possible arities is given by 2^N

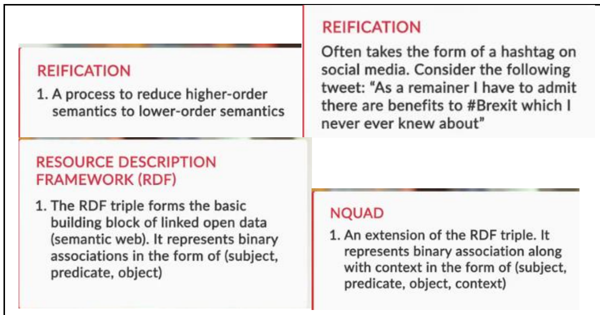
SEMANTIC ASSOCIATIONS

1. For practical semantic processing, you usually work with a maximum arity of 2
2. The resource description framework (RDF) triple forms the basic building block of linked open data (semantic web)
   a. Represents binary associations in the form of (subject, predicate, object)

Higher order predicates (i.e. having many entity types as parameters) are complex to deal with.

Even if all the binary relationships of a ternary relationship are true, it still does not imply that the ternary relationship will also be true. This implies that, in general, we cannot simply break down complex sentences (i.e. higher order predicates) into multiple lower-order predicates and verify their truth by verifying the lower-order predicates individually.

To solve this problem, we use a concept called **reification.**



REIFICATION

1. A process to reduce higher-order semantics to lower-order semantics

REIFICATION

Often takes the form of a hashtag on social media. Consider the following tweet: "As a remainer I have to admit there are benefits to #Brexit which I never ever knew about"

RESOURCE DESCRIPTION FRAMEWORK (RDF)

1. The RDF triple forms the basic building block of linked open data (semantic web). It represents binary associations in the form of (subject, predicate, object)

NQUAD

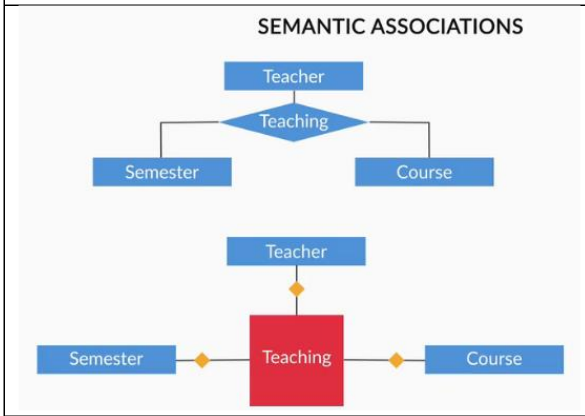
1. An extension of the RDF triple. It represents binary association along with context in the form of (subject, predicate, object, context)

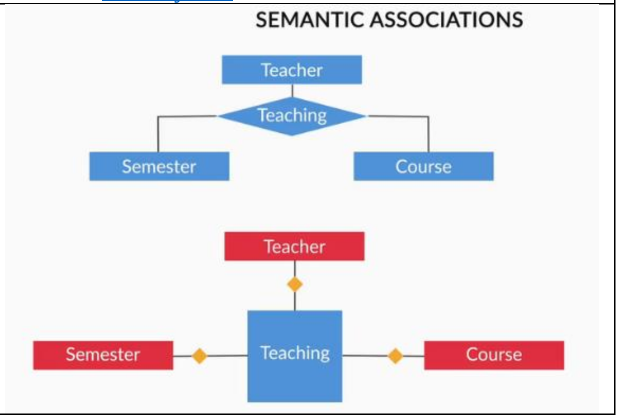
Reification refers to combining multiple entity types to convert them into lower order predicates.

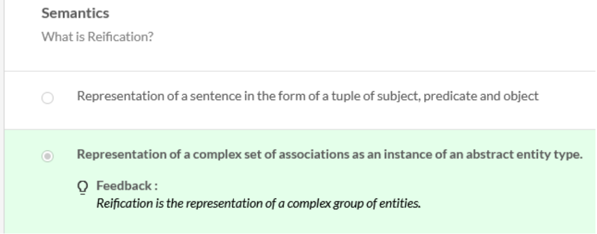
Next segment shows example of a huge knowledge graph **schema.org** which created a schema for a wide variety of entities (and reified entities) in one structured schema.

Additional Resources:
RDFs in detail

1. RDF-concepts
2. RDF-syntax







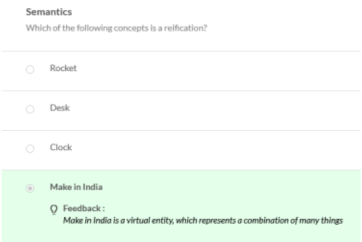
Semantics

What is Reification?

○ Representation of a sentence in the form of a tuple of subject, predicate and object

● Representation of a complex set of associations as an instance of an abstract entity type.

Feedback :
Reification is the representation of a complex group of entities.



Semantics

Which of the following concepts is a reification?

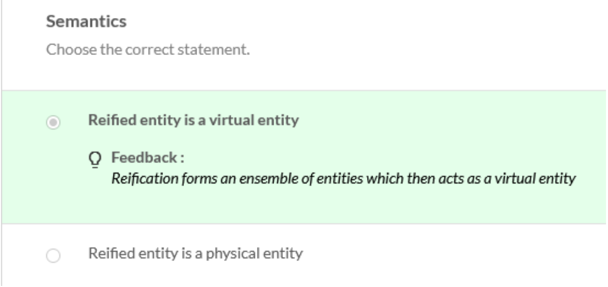
○ Rocket

○ Desk

○ Clock

● Make in India

Feedback :
Make in India is a virtual entity, which represents a combination of many things



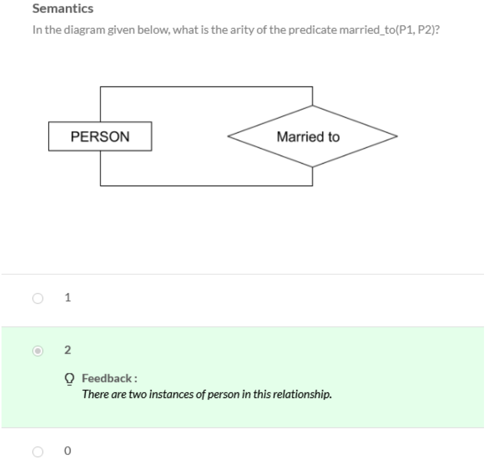
Semantics

Choose the correct statement.

● Reified entity is a virtual entity

Feedback :
Reification forms an ensemble of entities which then acts as a virtual entity

○ Reified entity is a physical entity



Semantics

In the diagram given below, what is the arity of the predicate married_to(P1, P2)?

○ 1

● 2

Feedback :
There are two instances of person in this relationship.

○ 0

Schema

We need a structure using which we can represent the meaning of sentences. One such schematic structure (used widely by search engines to index web pages) is **schema.org**, a joint effort by Google, Yahoo, Bing and Yandex to create a large schema relating the most commonly occurring entities on web pages. The main purpose of the schema is to ease search engine querying and improve search performance.

Say a web page of a hotel (e.g. Hotel Ginger) contains the words 'Ginger' and 'four stars'. How would a search engine indexing this page know whether the word 'Ginger' refers to the plant ginger or Hotel Ginger? Similarly, how would it know whether the phrase 'four stars' refers to the rating of a hotel or to astronomical stars?

To solve this problem, schema.org provides a way to explicitly specify the types of entities on web pages. For example, one can explicitly mention that 'Ginger' is the name of the hotel and specify various entities such as its rating, price etc. (example HTML shown below).

Attributes itemtype='Hotel' and itemprop='name', 'rating' and 'price' explicitly mention that the text on the web page (in the section inside the <div>) is about a hotel, that the hotel's name is Ginger, its rating is four stars and the price is INR 3500.

```
<div itemscope itemtype ="http://schema.org/Hotel">
  <h1 itemprop="name">Ginger</h1>
  <span>Rating: <span itemprop="rating">Four Star Hotel</span>
  <span itemprop="price">INR 3500</span>
</div>
```

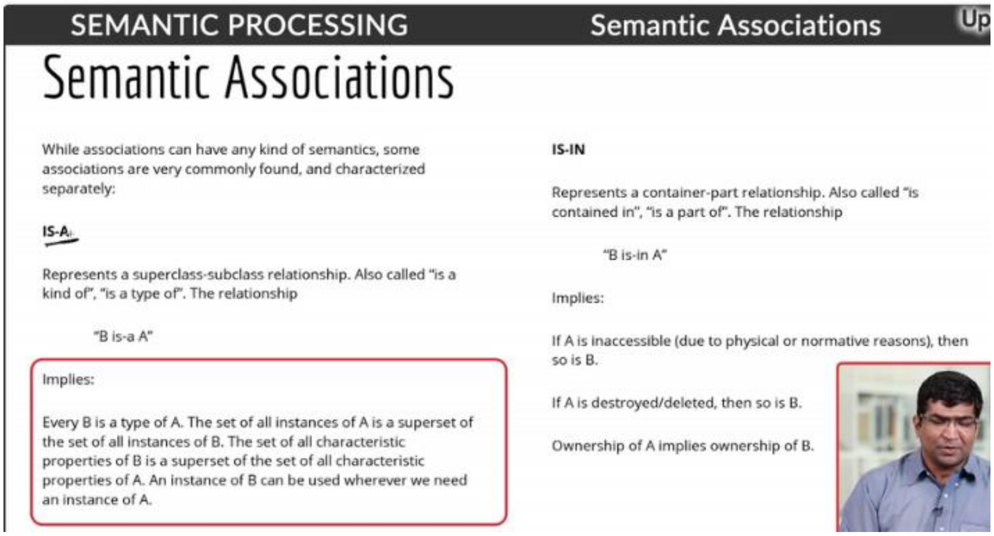
Huge schemas or 'knowledge graphs' of the world, such as schema.org, are quite important while building semantic processing engines (such as web search engines). Although you will not be working on such schemas/web-based search in this course, it is an important resource to know about.

Read more about schema.org here:

- An example showing how a web page can use schema.org to specify entity types
- How banks and financial institutions can use schemas for marking up banks and their products
- Schema.org FAQs: The basics of schema.org (who, what, why etc.) answered concisely

Semantic Associations

You have studied that entities have associations such as "a hotel has a price", "a hotel has a rating", "ginger is a plant" etc. Let's study some common types of associations.



SEMANTIC PROCESSING — Semantic Associations

Semantic Associations

While associations can have any kind of semantics, some associations are very commonly found, and characterized separately:

IS-A

Represents a superclass-subclass relationship. Also called "is a kind of", "is a type of". The relationship

"B is-a A"

Implies:

Every B is a type of A. The set of all instances of A is a superset of the set of all instances of B. The set of all characteristic properties of B is a superset of the set of all characteristic properties of A. An instance of B can be used wherever we need an instance of A.

IS-IN

Represents a container-part relationship. Also called "is contained in", "is a part of". The relationship

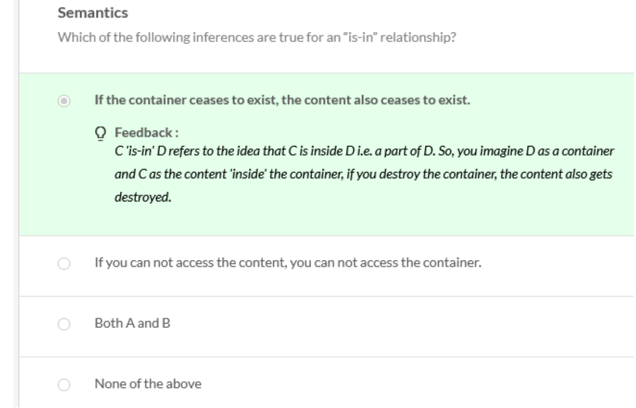
"B is-in A"

Implies:

If A is inaccessible (due to physical or normative reasons), then so is B.

If A is destroyed/deleted, then so is B.

Ownership of A implies ownership of B.



Semantics

Which of the following inferences are true for an "is-in" relationship?

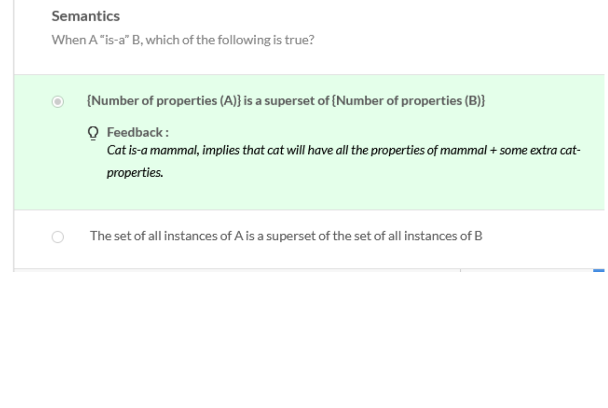
◉ If the container ceases to exist, the content also ceases to exist.

Feedback :
C 'is-in' D refers to the idea that C is inside D i.e. a part of D. So, you imagine D as a container and C as the content 'inside' the container, if you destroy the container, the content also gets destroyed.

○ If you can not access the content, you can not access the container.

○ Both A and B

○ None of the above



Semantics

When A "is-a" B, which of the following is true?

◉ {Number of properties (A)} is a superset of {Number of properties (B)}

Feedback :
Cat is-a mammal, implies that cat will have all the properties of mammal + some extra cat-properties.

○ The set of all instances of A is a superset of the set of all instances of B

Aboutness

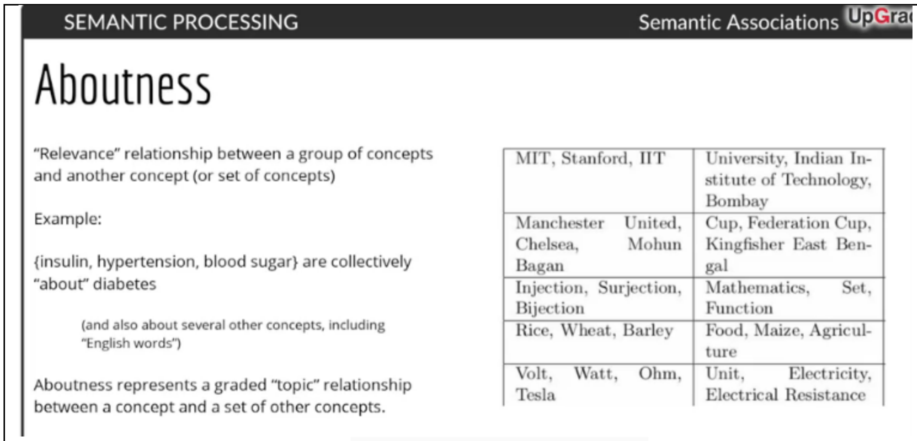
When machines are analysing text, we not only want to know the type of semantic associations 'is-a' and 'is-in' but also want to know what the word or sentence is about. Take, for example, below example

- *'Croatia fought hard before succumbing to France's deadly attack; lost the finals 2 goals to 4.'*

In the above text, if we want the machine to detect the game of football (it could be about other sports such as hockey as well, but let's keep things simple and assume it's about football), then we need to formally define the notion of **aboutness**.

We can, for example, detect that the game is football by defining semantic associations such as "Croatia" is-a "country", "France" is-a "country", "finals" is-a "tournament stage", "goals" is-a "scoring parameter" and so on. By defining such relationships, we can probably infer that the text is talking about football by going through the enormous schema. But imagine the kind of search this simple sentence would require. Even if we search through the schema, it doesn't mean we'll be able to decide that the game is football.

This leads us to define another semantic association - '**aboutness**'.



SEMANTIC PROCESSING Semantic Associations UpGrad

Aboutness

"Relevance" relationship between a group of concepts and another concept (or set of concepts)

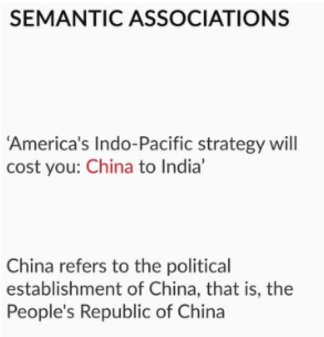
Example:

{insulin, hypertension, blood sugar} are collectively "about" diabetes

(and also about several other concepts, including "English words")

Aboutness represents a graded "topic" relationship between a concept and a set of other concepts.

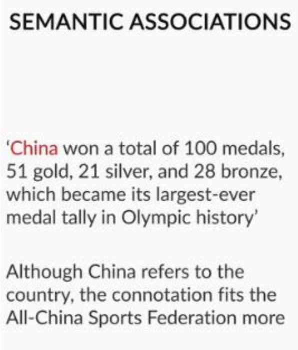
MIT, Stanford, IIT	University, Indian Institute of Technology, Bombay
Manchester United, Chelsea, Mohun Bagan	Cup, Federation Cup, Kingfisher East Bengal
Injection, Surjection, Bijection	Mathematics, Set, Function
Rice, Wheat, Barley	Food, Maize, Agriculture
Volt, Watt, Ohm, Tesla	Unit, Electricity, Electrical Resistance



SEMANTIC ASSOCIATIONS

'America's Indo-Pacific strategy will cost you: China to India'

China refers to the political establishment of China, that is, the People's Republic of China



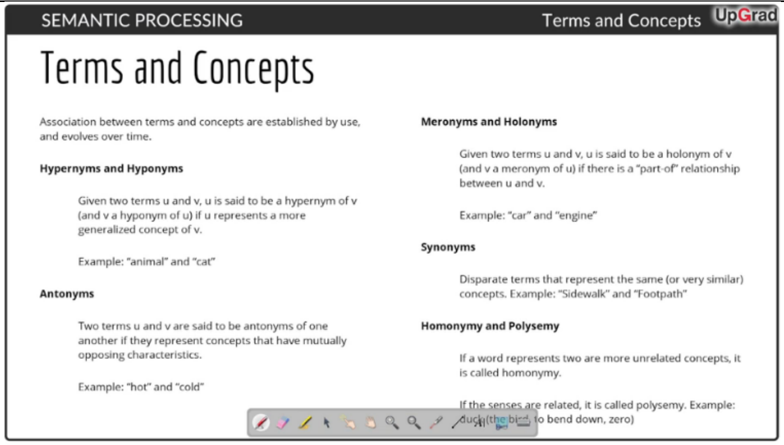
SEMANTIC ASSOCIATIONS

'China won a total of 100 medals, 51 gold, 21 silver, and 28 bronze, which became its largest-ever medal tally in Olympic history'

Although China refers to the country, the connotation fits the All-China Sports Federation more

To understand the 'aboutness' of a text means to identify the 'topics' being talked about in the text. What makes this problem hard is that the same word (e.g. China) can be used in multiple topics such as politics, the Olympic games, trading etc.

Idea of 'aboutness' and topics in detail in the third session on **topic modelling**. For now, let's study some nomenclatures used to classify types of associations between terms and concepts.



SEMANTIC PROCESSING Terms and Concepts UpGrad

Terms and Concepts

Association between terms and concepts are established by use, and evolves over time.

Hypernyms and Hyponyms

Given two terms u and v, u is said to be a hypernym of v (and v a hyponym of u) if u represents a more generalized concept of v.

Example: "animal" and "cat"

Antonyms

Two terms u and v are said to be antonyms of one another if they represent concepts that have mutually opposing characteristics.

Example: "hot" and "cold"

Meronyms and Holonyms

Given two terms u and v, u is said to be a holonym of v (and v a meronym of u) if there is a "part-of" relationship between u and v.

Example: "car" and "engine"

Synonyms

Disparate terms that represent the same (or very similar) concepts. Example: "Sidewalk" and "Footpath"

Homonymy and Polysemy

If a word represents two are more unrelated concepts, it is called homonymy.

If the senses are related, it is called polysemy. Example: duck (the bird, to bend down, zero)

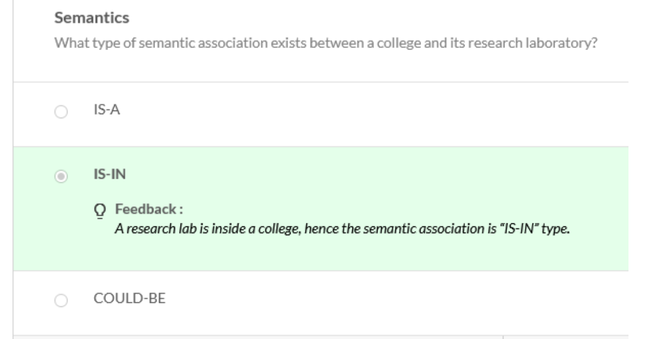
Homonymy and **polysemy**: Words having different meanings but the same spelling and pronunciations are called homonyms. For example, the word 'bark' in 'dog's bark' is a homonym to the word 'bark' in 'bark of a tree'. Polysemy is when a word has multiple (entirely different) meanings. For example, consider the word 'pupil'. It can either refer to students or eye pupil, depending upon the context in which it is used.

Hypernyms and **hyponyms**: This shows the relationship between a generic term (hypernym) and a specific instance of it (hyponym). For example, the term 'Punjab National Bank' is a hyponym of the generic term 'bank'

Antonyms: Words that are opposite in meanings are said to be antonyms of each other. Example hot and cold, black and white etc.

Meronyms and **Holonyms**: A term 'A' is said to be a holonym of term 'B' if 'B is part of 'A' (while the term 'B' is said to be a meronym of the term 'A'). For example, an operating system is part of a computer. Here, 'computer' is the holonym of 'operating system' whereas 'operating system' is the meronym of 'computer.

Synonyms: Terms that have a similar meaning are synonyms to each other. For example, 'glad' and 'happy'.



Semantics

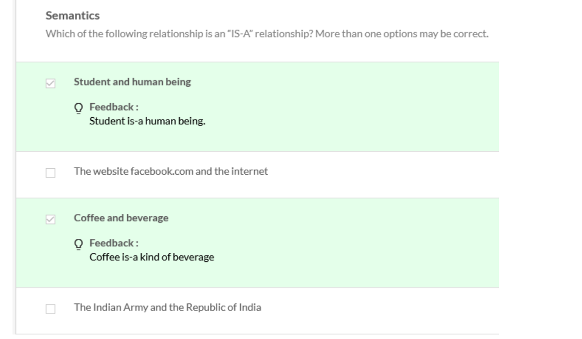
What type of semantic association exists between a college and its research laboratory?

○ IS-A

● **IS-IN**

Feedback :
A research lab is inside a college, hence the semantic association is "IS-IN" type.

○ COULD-BE



Semantics

Which of the following relationship is an "IS-A" relationship? More than one options may be correct.

☑ **Student and human being**

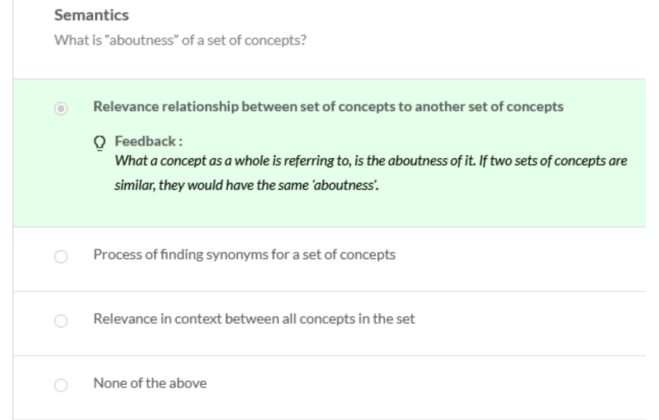
Feedback :
Student is-a human being.

□ The website facebook.com and the internet

☑ **Coffee and beverage**

Feedback :
Coffee is-a kind of beverage

□ The Indian Army and the Republic of India



Semantics

What is "aboutness" of a set of concepts?

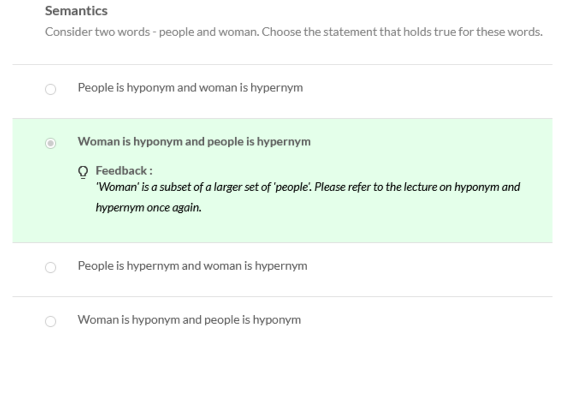
● **Relevance relationship between set of concepts to another set of concepts**

Feedback :
What a concept as a whole is referring to, is the aboutness of it. If two sets of concepts are similar, they would have the same 'aboutness'.

○ Process of finding synonyms for a set of concepts

○ Relevance in context between all concepts in the set

○ None of the above



Semantics

Consider two words - people and woman. Choose the statement that holds true for these words.

○ People is hyponym and woman is hypernym

● **Woman is hyponym and people is hypernym**

Feedback :
'Woman' is a subset of a larger set of 'people'. Please refer to the lecture on hyponym and hypernym once again.

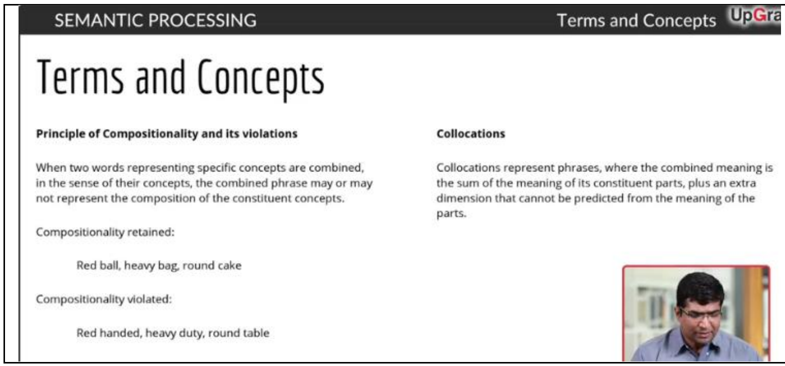
○ People is hypernym and woman is hypernym

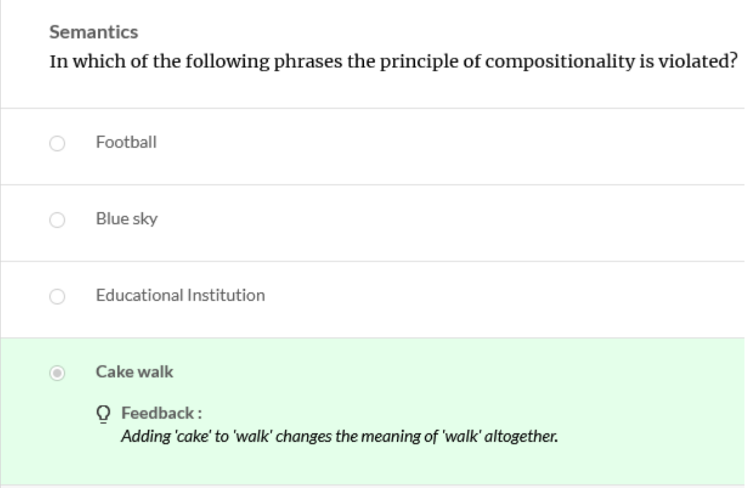
○ Woman is hyponym and people is hyponym

Even after defining such a wide range of association types, one cannot cover the wide range of complexities of natural languages.

For example, consider how two words are often put together to form a phrase. The semantics of the combination of these words could be very different than the individual words. For example, consider the phrase - ‘cake walk’. The meanings of the terms 'cake' and 'walk' are very different from the meaning of their combination.

Such cases are said to violate the **principle of compositionality.**

 SEMANTIC PROCESSING Terms and Concepts Terms and Concepts **Principle of Compositionality and its violations** When two words representing specific concepts are combined, in the sense of their concepts, the combined phrase may or may not represent the composition of the constituent concepts. Compositionality retained: Red ball, heavy bag, round cake Compositionality violated: Red handed, heavy duty, round table **Collocations** Collocations represent phrases, where the combined meaning is the sum of the meaning of its constituent parts, plus an extra dimension that cannot be predicted from the meaning of the parts.	Principle of compositionality, although valid in most cases, is often violated as well. This is an important insight to understand the nature of semantics and will be useful in developing techniques and algorithms for semantic processing.



Semantics

In which of the following phrases the principle of compositionality is violated?

○ Football

○ Blue sky

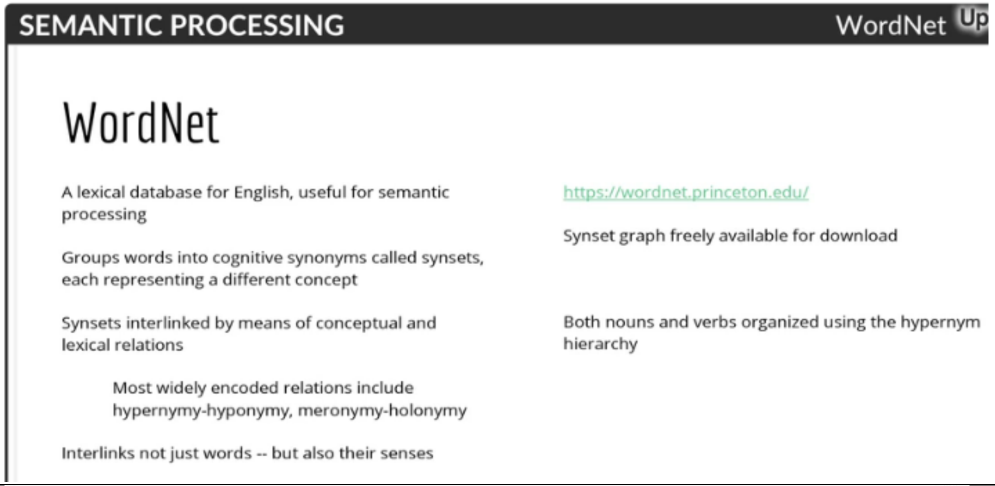
○ Educational Institution

● Cake walk

Feedback :
Adding 'cake' to 'walk' changes the meaning of 'walk' altogether.

Databases - WordNet and ConceptNet

WordNet is a semantically oriented dictionary of English, like a traditional thesaurus but with a richer structure.



SEMANTIC PROCESSING — WordNet

WordNet

A lexical database for English, useful for semantic processing

Groups words into cognitive synonyms called synsets, each representing a different concept

Synsets interlinked by means of conceptual and lexical relations

Most widely encoded relations include hypernymy-hyponymy, meronymy-holonymy

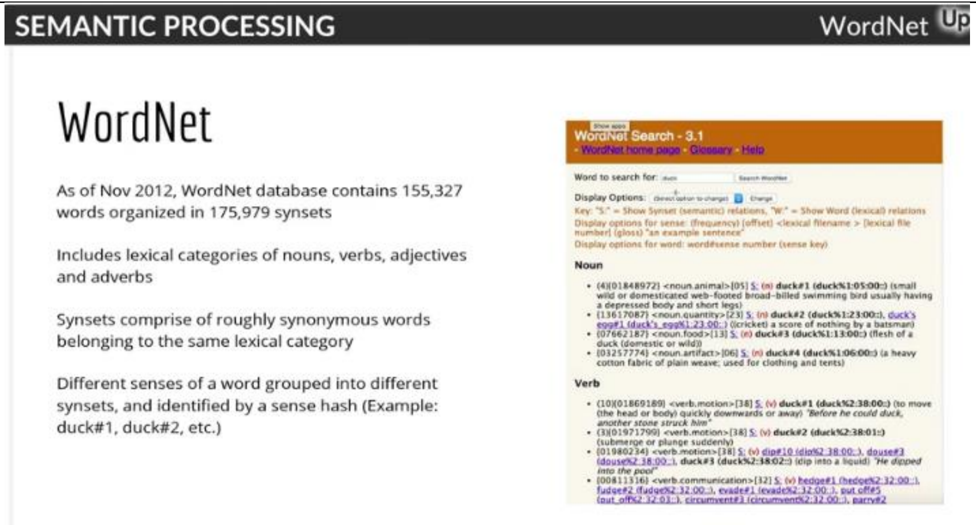
Interlinks not just words -- but also their senses

https://wordnet.princeton.edu/

Synset graph freely available for download

Both nouns and verbs organized using the hypernym hierarchy

WordNet is a part of NLTK and you will use WordNet later in this module to identify the 'correct' sense of a word (i.e for word sense disambiguation).



SEMANTIC PROCESSING — WordNet

WordNet

As of Nov 2012, WordNet database contains 155,327 words organized in 175,979 synsets

Includes lexical categories of nouns, verbs, adjectives and adverbs

Synsets comprise of roughly synonymous words belonging to the same lexical category

Different senses of a word grouped into different synsets, and identified by a sense hash (Example: duck#1, duck#2, etc.)

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: duck

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (frequency) {offset} <lexical filename> [lexical file number] (gloss) "an example sentence"

Display options for word: word#sense number (sense key)

Noun

- (4){01848972} <noun.animal>[05] S: (n) **duck#1 (duck%1:05:00::)** (small wild or domesticated web-footed broad-billed swimming bird usually having a depressed body and short legs)
- {13617087} <noun.quantity>[23] S: (n) **duck#2 (duck%1:23:00::)**, duck's egg#1 (duck's_egg%1:23:00::) ((cricket) a score of nothing by a batsman)
- {07662187} <noun.food>[13] S: (n) **duck#3 (duck%1:13:00::)** (flesh of a duck (domestic or wild))
- {03257774} <noun.artifact>[06] S: (n) **duck#4 (duck%1:06:00::)** (a heavy cotton fabric of plain weave; used for clothing and tents)

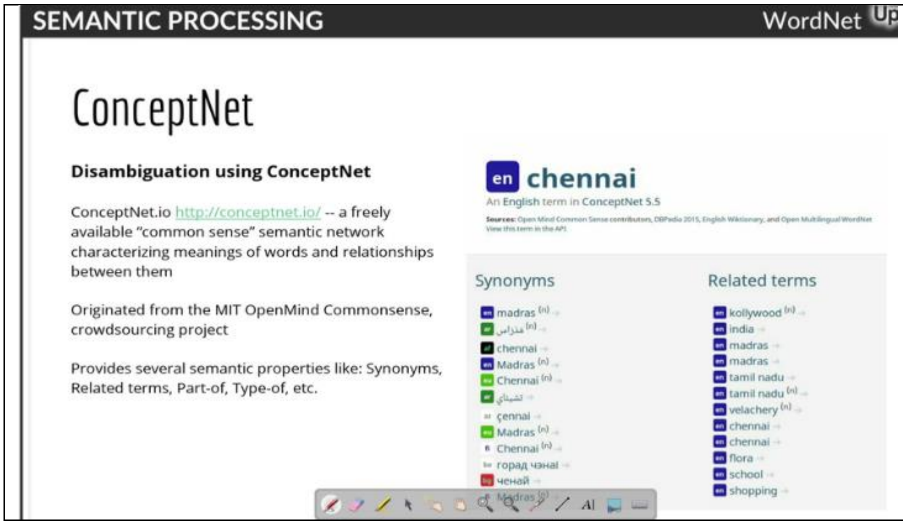
Verb

- (10){01869189} <verb.motion>[38] S: (v) **duck#1 (duck%2:38:00::)** (to move (the head or body) quickly downwards or away) *"Before he could duck, another stone struck him"*
- (3){01971799} <verb.motion>[38] S: (v) **duck#2 (duck%2:38:01::)** (submerge or plunge suddenly)
- {01980234} <verb.motion>[38] S: (v) dip#10 (dip%2:38:00::), douse#3 (douse%2:38:00::), **duck#3 (duck%2:38:02::)** (dip into a liquid) *"He dipped into the pool"*
- {00811316} <verb.communication>[32] S: (v) hedge#1 (hedge%2:32:00::), fudge#2 (fudge%2:32:00::), evade#1 (evade%2:32:00::), put off#5 (put_off%2:32:03::), circumvent#3 (circumvent%2:32:00::), parry#2

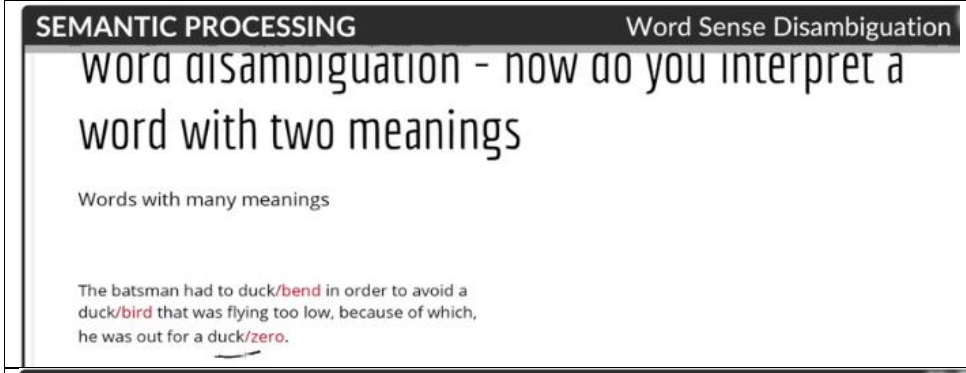
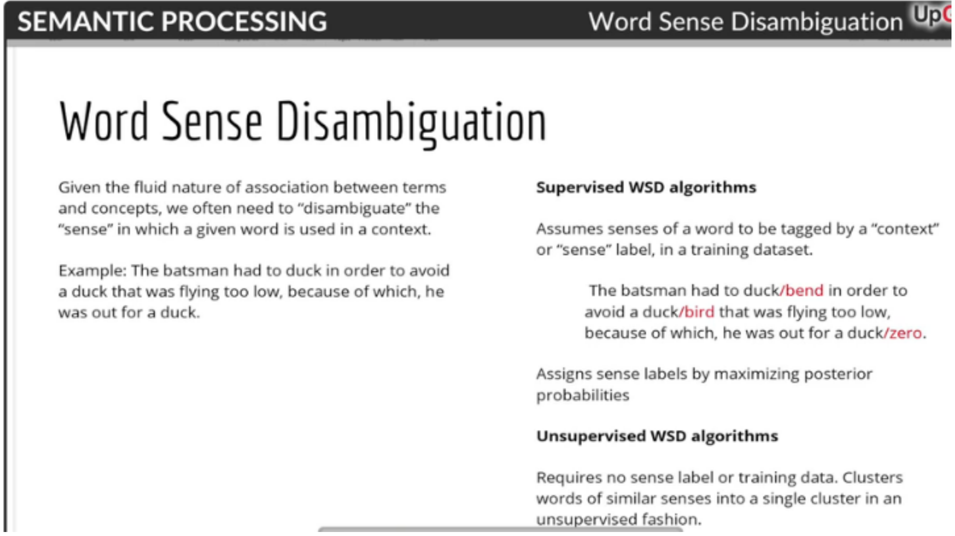
Additional Readings:

1. Wordnet
2. Tutorial on Wordnet - Python
3. Conceptnet

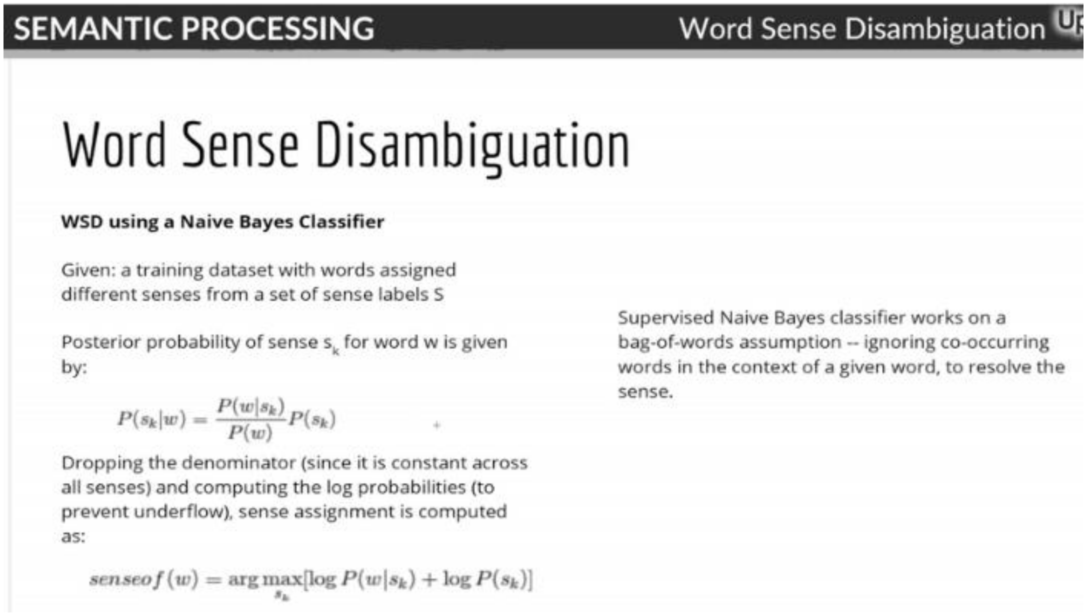
Another important resource for semantic processing is **ConceptNet** which deals specifically with assertions between concepts. For example, there is the concept of a “dog”, and the concept of a “kennel”. As a human, we know that a dog lives inside a kennel. ConceptNet records that assertion with /c/en/**dog** /r/**AtLocation** /c/en/**kennel**.	ConceptNet is a representation that provides commonsense linkages between words. For example, it states that bread is commonly found near toasters. These everyday facts could be useful if, for e.g., to make a smart chatbot which says - “Since you like toasters, do also like bread? I can order some for you.”

 SEMANTIC PROCESSING WordNet Up ConceptNet **Disambiguation using ConceptNet** ConceptNet.io http://conceptnet.io/ -- a freely available "common sense" semantic network characterizing meanings of words and relationships between them Originated from the MIT OpenMind Commonsense, crowdsourcing project Provides several semantic properties like: Synonyms, Related terms, Part-of, Type-of, etc. en chennai An English term in ConceptNet 5.5 Sources: Open Mind Common Sense contributors, DBPedia 2015, English Wiktionary, and Open Multilingual WordNet View this term in the API Synonyms en madras (n) ar مدراس (n) zh chennai en Madras (n) eu Chennai (n) ar تشيناي az çennai eu Madras (n) fi Chennai (n) be горад чэнаі bg ченай Madras (n) Related terms en kollywood (n) en india en madras en madras en tamil nadu en tamil nadu (n) en velachery (n) en chennai en chennai en flora en school en shopping	It isn't organized as well as one would want. For instance, it explicitly states that a toaster is related to an automobile. This is true since they are both mechanical machines, but you wouldn't want for e.g. a chatbot to learn that relationship in most contexts.

Word Sense Disambiguation - Naive Bayes

 SEMANTIC PROCESSING Word Sense Disambiguation word disambiguation - how do you interpret a word with two meanings Words with many meanings The batsman had to duck/bend in order to avoid a duck/bird that was flying too low, because of which, he was out for a duck/zero.	A common problem in semantic analysis - **word sense disambiguation** (WSD) is the task of identifying the correct sense of an ambiguous word such as 'bank', 'bark', 'pitch' etc.
 SEMANTIC PROCESSING Word Sense Disambiguation Word Sense Disambiguation Given the fluid nature of association between terms and concepts, we often need to "disambiguate" the "sense" in which a given word is used in a context. Example: The batsman had to duck in order to avoid a duck that was flying too low, because of which, he was out for a duck. **Supervised WSD algorithms** Assumes senses of a word to be tagged by a "context" or "sense" label, in a training dataset. The batsman had to duck/bend in order to avoid a duck/bird that was flying too low, because of which, he was out for a duck/zero. Assigns sense labels by maximizing posterior probabilities **Unsupervised WSD algorithms** Requires no sense label or training data. Clusters words of similar senses into a single cluster in an unsupervised fashion.	WSD is basically a tagging problem. **Supervised techniques** for WSD require the input words to be tagged with their senses. The sense is the label assigned to the word. In **unsupervised techniques**, words are not tagged with their senses, which are to be inferred using other techniques.

In supervised techniques, one of the simplest text classification algorithms is the **Naive Bayes Classifier.**



SEMANTIC PROCESSING

Word Sense Disambiguation

Word Sense Disambiguation

WSD using a Naive Bayes Classifier

Given: a training dataset with words assigned different senses from a set of sense labels S

Posterior probability of sense s_k for word w is given by:

$$P(s_k|w) = \frac{P(w|s_k)}{P(w)} P(s_k)$$

Dropping the denominator (since it is constant across all senses) and computing the log probabilities (to prevent underflow), sense assignment is computed as:

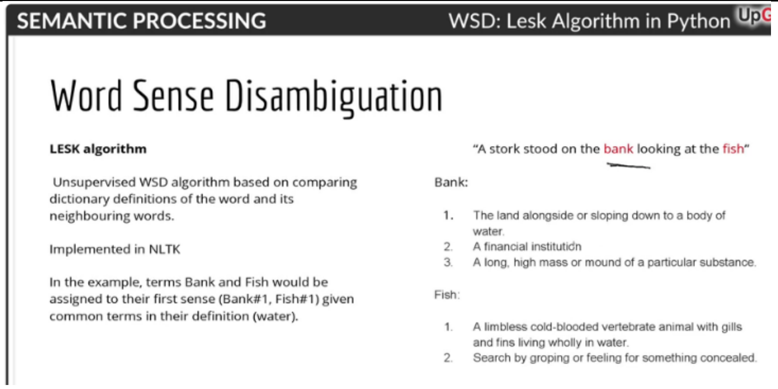
$$senseof(w) = \arg\max_{s_k}[\log P(w|s_k) + \log P(s_k)]$$

Supervised Naive Bayes classifier works on a bag-of-words assumption -- ignoring co-occurring words in the context of a given word, to resolve the sense.

Word Sense Disambiguation - Lesk Algorithm

As with most machine learning problems, lack of labelled data is a common problem in building disambiguation models. Thus, we need unsupervised techniques to solve the same problem.

A popular unsupervised algorithm used for word sense disambiguation is the **Lesk algorithm**.



SEMANTIC PROCESSING

WSD: Lesk Algorithm in Python

Word Sense Disambiguation

LESK algorithm

Unsupervised WSD algorithm based on comparing dictionary definitions of the word and its neighbouring words.

Implemented in NLTK

In the example, terms Bank and Fish would be assigned to their first sense (Bank#1, Fish#1) given common terms in their definition (water).

"A stork stood on the bank looking at the fish"

Bank:

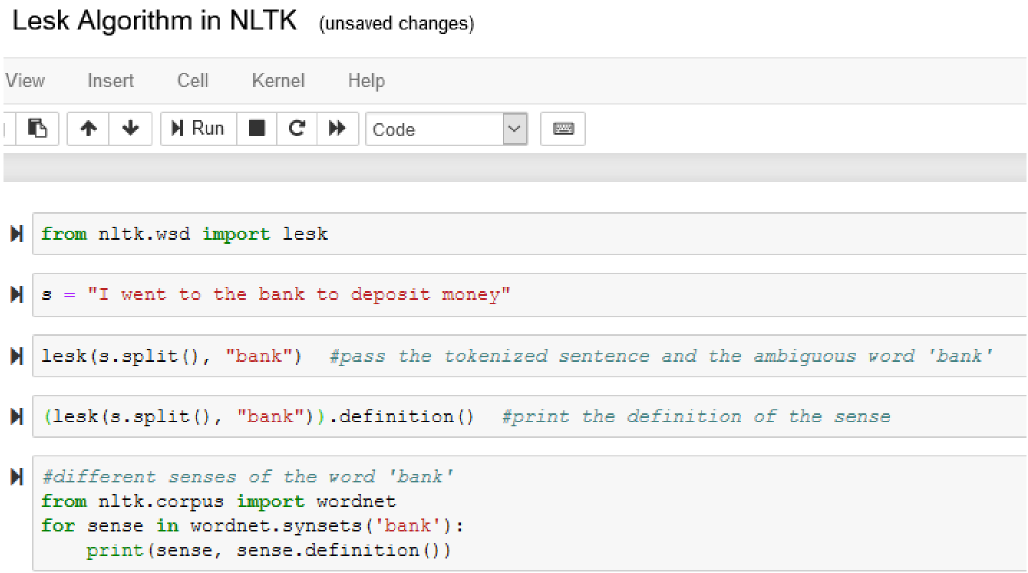
1. The land alongside or sloping down to a body of water.
2. A financial institution
3. A long, high mass or mound of a particular substance.

Fish:

1. A limbless cold-blooded vertebrate animal with gills and fins living wholly in water.
2. Search by groping or feeling for something concealed.

Various ways to use the lesk algorithm. Apart from what the professor has discussed, take the definitions corresponding to the different senses of the ambiguous word and see which definition overlaps maximum with the neighbouring words of the ambiguous word. The sense which has the maximum overlap with the surrounding words is then chosen as the 'correct sense'.

Let's now use lesk algorithm to disambiguate the word '**bank**' in a text.



Lesk Algorithm in NLTK (unsaved changes)

```python
from nltk.wsd import lesk
```

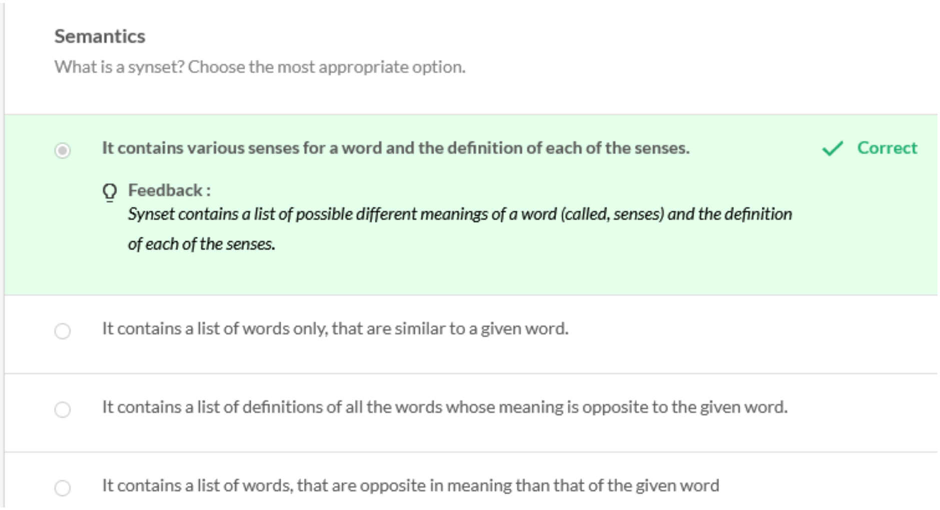
```python
s = "I went to the bank to deposit money"
```

```python
lesk(s.split(), "bank")  #pass the tokenized sentence and the ambiguous word 'bank'
```

```python
(lesk(s.split(), "bank")).definition()  #print the definition of the sense
```

```python
#different senses of the word 'bank'
from nltk.corpus import wordnet
for sense in wordnet.synsets('bank'):
    print(sense, sense.definition())
```

WordNet has a network of synonyms called synset for individual words.



Semantics

What is a synset? Choose the most appropriate option.

● **It contains various senses for a word and the definition of each of the senses.** ✓ Correct

Feedback :
Synset contains a list of possible different meanings of a word (called, senses) and the definition of each of the senses.

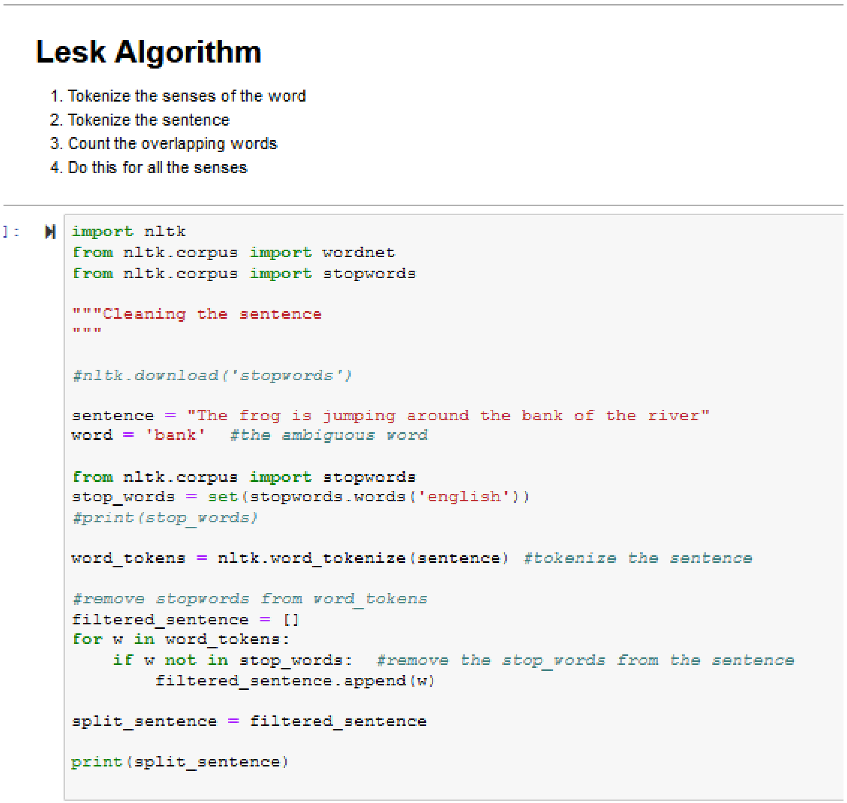
○ It contains a list of words only, that are similar to a given word.

○ It contains a list of definitions of all the words whose meaning is opposite to the given word.

○ It contains a list of words, that are opposite in meaning than that of the given word

Lesk Algorithm Implementation

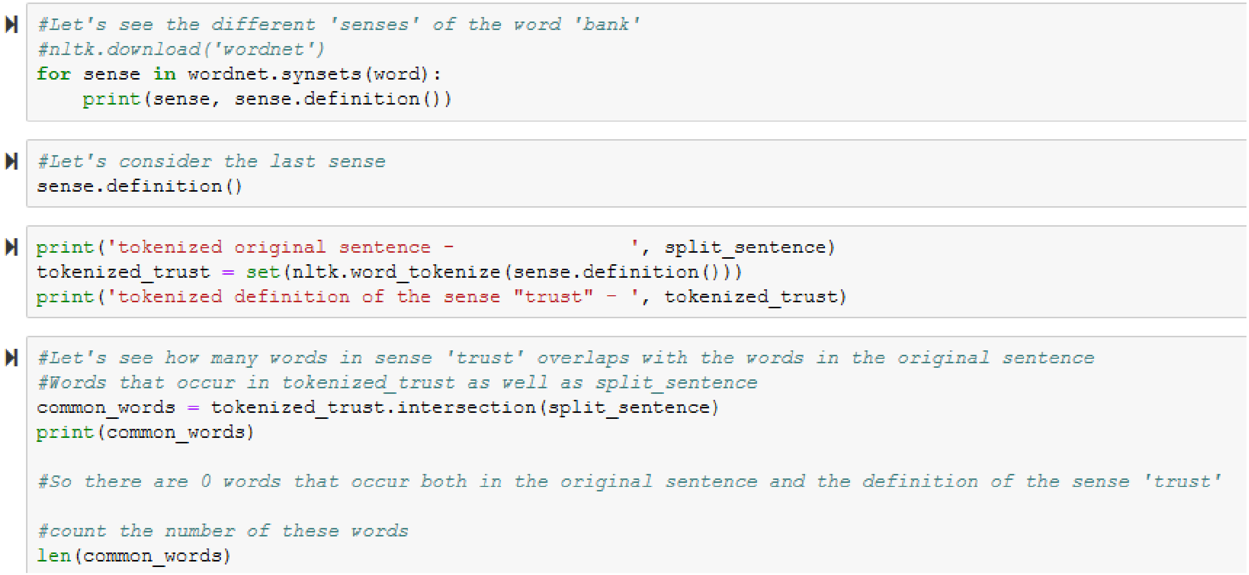
Let’s implement the lesk algorithm in python from scratch in this segment, unlike the previous section where we used NLTK’s implementation of lesk algorithm.



Lesk Algorithm

1. Tokenize the senses of the word
2. Tokenize the sentence
3. Count the overlapping words
4. Do this for all the senses

```
import nltk
from nltk.corpus import wordnet
from nltk.corpus import stopwords

"""Cleaning the sentence
"""

#nltk.download('stopwords')

sentence = "The frog is jumping around the bank of the river"
word = 'bank'  #the ambiguous word

from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#print(stop_words)

word_tokens = nltk.word_tokenize(sentence) #tokenize the sentence

#remove stopwords from word_tokens
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:  #remove the stop_words from the sentence
        filtered_sentence.append(w)

split_sentence = filtered_sentence

print(split_sentence)
```

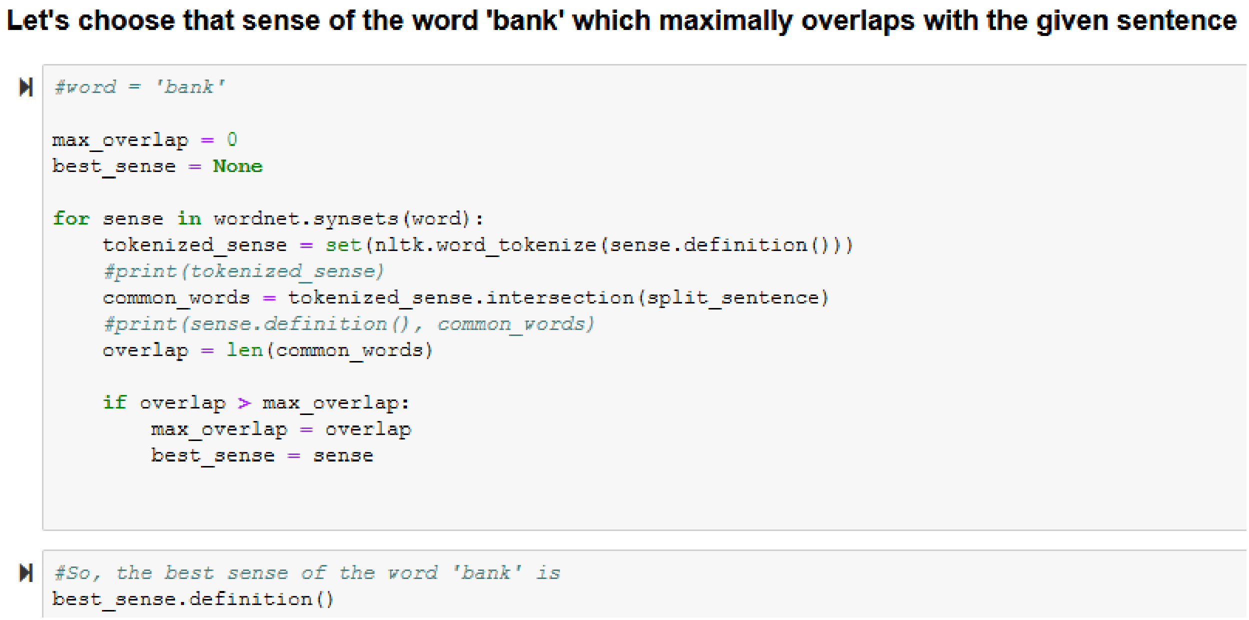


```
#Let's see the different 'senses' of the word 'bank'
#nltk.download('wordnet')
for sense in wordnet.synsets(word):
    print(sense, sense.definition())
```

```
#Let's consider the last sense
sense.definition()
```

```
print('tokenized original sentence -                ', split_sentence)
tokenized_trust = set(nltk.word_tokenize(sense.definition()))
print('tokenized definition of the sense "trust" - ', tokenized_trust)
```

```
#Let's see how many words in sense 'trust' overlaps with the words in the original sentence
#Words that occur in tokenized_trust as well as split_sentence
common_words = tokenized_trust.intersection(split_sentence)
print(common_words)

#So there are 0 words that occur both in the original sentence and the definition of the sense 'trust'

#count the number of these words
len(common_words)
```



Let's choose that sense of the word 'bank' which maximally overlaps with the given sentence

```
#word = 'bank'

max_overlap = 0
best_sense = None

for sense in wordnet.synsets(word):
    tokenized_sense = set(nltk.word_tokenize(sense.definition()))
    #print(tokenized_sense)
    common_words = tokenized_sense.intersection(split_sentence)
    #print(sense.definition(), common_words)
    overlap = len(common_words)

    if overlap > max_overlap:
        max_overlap = overlap
        best_sense = sense
```

```
#So, the best sense of the word 'bank' is
best_sense.definition()
```

Lesk algorithm helps in word sense disambiguation. The word '**bank** 'can have multiple meanings depending on the surrounding (or the context) words. The lesk algorithm helps in finding the 'correct' meaning.

Summary

Basic ideas to represent meaning - **entities, entity types, arity, reification** and various types of **semantic associations** that can exist between entities.

Idea of aboutness - text is always about something, and there are techniques to infer the topics the text is about.

Associations between a wide range of entities are stored in a structured way in gigantic knowledge graphs or schemas such as **schema.org**.

Techniques that can be used to **disambiguate** the meaning of a word - supervised and unsupervised. The 'correct' meaning of an ambiguous word depends upon the contextual words.

In supervised techniques, such as **naive Bayes** (or any classifier for that matter), take the context-sense set as the training data. The label is the 'sense' and the input is the context words.

In unsupervised techniques, such as the **lesk algorithm**, assign the definition to the ambiguous word which overlaps with the surrounding words maximally.