1) Porter stemmer( Word is given and find out it's stemmed word using porter stemmer)(Numerical)

2) CVC Combination

3) Find out measure of the word

4) Explain Porter Stemmer (sb rules explain krne hote hai (5M), ek do rules likho and flowchart Karo)

* **Stemming** is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. For example : words such as "Likes", "liked", " likely" and "liking" will be reduced to "like" after stemming.
* The process of determining the root of words is known as stemming.

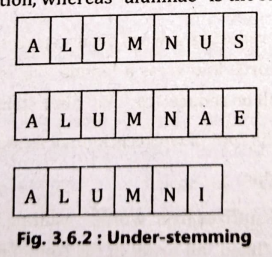
**(A) Over-stemming**

* Over-stemming is the term we use to describe the process through which our algorithm stems many unrelated words to the same root.
* Despite sharing a core word and being related, the words "universal," "university," and "universe" have quite different meanings.
* These terms should return quite distinct search results when we enter them into a reputable search engine, and they shouldn't be considered synonyms. Such a mistake is known as a false positive.



**(B) Under-stemming**

The converse of that phenomenon, known as under-Stemming, occurs when numerous words are not stemmed from a single root even when they should. A former university student is referred to as a "alumnus" and is often a male individual. The word "alumni" refers to several former students of an institution, whereas "alumnae" is the feminine equivalent.



**Algorithm:**

To present the suffix stripping algorithm in its entirety we will need a few definitions. A consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. So, in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A consonant will be denoted by c, a vowel by v. A list ccc ... of length greater than 0 will be denoted by C, and a list vvv ... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms :

* CVCV ... C
* CVCV ... V
* VCVC ... C
* VCVC , .. V

These may all be represented by the single form

[C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents. Using (VC)m to denote VC repeated m times, this may again be written as

[C] (VC)m [V]

m will be called the measure of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples :

* m=0 OTR, EE, TREE, Y, BY.
* m=1 TROUBLE, OATS, TREES, IVY.
* m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

**Rule for removing a suffix**

* The rules are of the form: (condition) S1 -> S2 Where S1 and S2 are suffixes.
* This means that if a word ends with a suffix S1, and the stem before S1 satisfies the given condition then S1 is replaced by S2.

eg (m>1) EMENT ->

* Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

The 'condition' part may also contain the following :

|  |  |
| --- | --- |
| m | The measure of the stem |
| \*S | The stem ends with S |
| \*v\* | The stem contains a vowel |
| \*d | The stem ends with a double consonant (TT, SS) |
| \*o | The stem ends in CVC (second C not W, X, or Y)  E.g. WIL, HOP |

And the condition part may also contain expressions with and, or and not, so that :

(m>1 and (\*S or \*T)) tests for a stem with m>1 ending in S or T, while (\*d and not (\*L or \*S or \*Z)) tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely.

**Step 1a :**

* SSES -> SS (Example : caresses -> caress)
* IES -> I (Example : ponies -> poni ; ties -> ti)
* SS -> SS (Example : caress -> caress)
* S -> € (Example : cats -> cat)
* SSES -> SS: This rule reduces words ending in "sses" to just "ss". It's primarily aimed at plural nouns. Example: Input: caresses, Output: caress
* IES -> I: This rule changes words ending in "ies" to end in just "i". It's also primarily focused on plural nouns. Example: Input: ponies, Output: poni
  + SS -> SS: This part is more of a placeholder in the algorithm. It doesn't change the word but serves as a reference point.
  + S -> (remove the "s" suffix): This part removes the trailing "s" from words, primarily aimed at singular and plural nouns. Example: Input: cats, Output: cat

Rule 1a helps normalize words by reducing them to their base forms, especially in cases where pluralization or verb conjugation adds extra letters that aren't part of the word's core meaning.

**Step 1b:**

* + (m>1) EED -> EE

Condition verified: agreed -> agree

Condition not verified: needed -> need

* + (\*V\*) ED -> €

Condition verified: plastered -> plaster

Condition not verified: bled -> bled

* + (\*V\*) ING -> €

Condition verified: motoring -> motor

Condition not verified: sing -> sing

Rule 1b of the Porter Stemmer algorithm addresses additional suffixes beyond those covered in Rule 1a. Here's an explanation of each part of Rule 1b with examples:

* 1. **EED or ED -> EE**: This rule checks for words ending in "eed" or "ed" and replaces them with "ee" or simply removes the "ed" suffix. It primarily targets past tense verbs.

Input: needed, Output: need

* 1. **ING ->** (remove the "ing" suffix): 2. This part removes the "-ing" suffix from words, typically indicating a present participle or gerund.

Example: Input: playing, Output: play

* 1. **ATIONAL -> ATE:** This rule converts words ending in "ational" to end in "ate". It's focused on transforming adjectives or nouns into verbs.

Example: Input: relational, Output: relate

* 1. **TIONAL -> TION:** Similar to the previous rule, this one converts words ending in "tional" to end in "tion".

Example: Input: conditional, Output: condition

These examples illustrate how Rule 1b of the Porter Stemmer algorithm applies various transformations to reduce words to their base forms by removing or replacing specific suffixes.

If the second or third of the rules in Step 1b is successful, the following is done:

* + AT-> ATE (Example : conflat(ed) -> conflate)
  + BL-> BLE (Example : troubl(ed) -> trouble)
  + IZ-> IZE (Example : siz(ed) -> size)
  + S-> (Example : cats-> cat)

(\*d and not (\*L or \*S or \*Z)) -> single letter

(Example : hopp(ing) -> hop ; tann(ed) -> tan ; fall(ing) -> fall ; hiss(ing) -> hiss ; fizz(ed) -> fizz)

(m=1 and \*o)-> E (Example : fail(ing)-> fail ; fil(ing)-> file)

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognised later. This E may be removed in step 4.

**Step 1c:**

(\*v\*) Y -> I (Example : happy -> happi ; sky -> sky)

* Step 1c of the Porter Stemmer algorithm aims to handle certain cases where the algorithm needs to address the removal of an "e" suffix from the end of a word.
* E -> (remove the "e" suffix if preceded by a short syllable):
* This rule checks if a word ends with an "e" and whether the preceding part of the word constitutes a "short syllable." A short syllable, in this context, generally refers to a syllable with a vowel followed by a consonant (VC). If these conditions are met, the final "e" is removed from the word.
* Example: Input: loveable

"lo" is considered a short syllable because it's followed by a consonant "v". Hence, the "e" at the end is removed.

Output: lovab

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

**Step 2**: Derivational Morphology, I

* (m>0) ATIONAL -> ATE, Relational -> relate
* (m>0) IZATION -> IZE, generalization-> generalize
* (m>0) BILITI -> BLE, sensibiliti -> sensible

**Step 3:** Derivational Morphology, II

* (m>0) ICATE -> IC, triplicate -> triplic
* (m>0) FUL -> €, hopeful -> hope
* (m>0) NESS -> E, goodness -> good

**Step 4:** Derivational Morphology, III

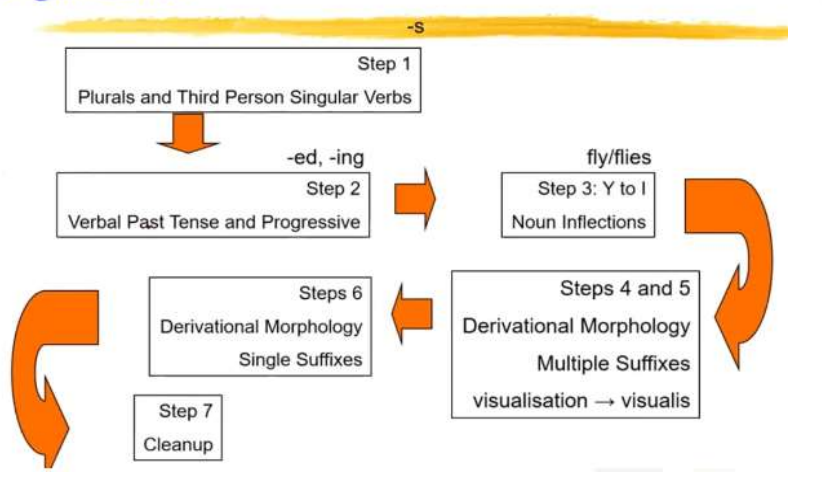
* (m>0) ANCE -> €, allowance-> allow
* (m>0) ENT -> €, dependent-> depend

**Step 5a**

* (m>1) E -> €, probate -> probat
* (m=1 & !\* o) NESS -> €, goodness -> good

**Step 5b**

* (m>1 & \*d & "L) -> single letter
* Condition verified: controll -> control
* Condition not verified: roll -> roll



5) Edit Distance (small string - 2m, big word - 5m)

* In Natural Language Processing (NLP), the edit distance is a measure of how different two strings of characters are from each other.
* It is the minimum number of single-character insertions, deletions, or substitutions needed to transform one string into the other.
* The edit distance is used in a variety of NLP applications, such as spellchecking, auto-correction, and machine translation.
* For example, in spell-checking, the edit distance is used to suggest corrections for misspelled words. In machine translation, the edit distance is used to align source and target language sentences and to identify the best translation.
* The algorithm for calculating edit distance involves building a matrix that represents the number of edit operations required to transform one string into another.
* The matrix is initialized with values that represent the cost of each operation (insertion, deletion, or substitution), and then filled in using dynamic programming.
* The edit distance can be calculated efficiently using dynamic programming algorithms such as the Wagner-Fisher algorithm or the Hirschberg algorithm.
* These algorithms have a time complexity of O(mn), where m and n are the lengths of the two strings being compared.
* Let's say we have two strings:

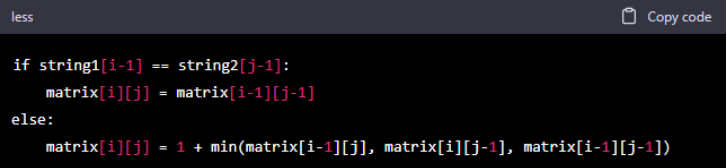


* To find the edit distance between these two strings, we can use a dynamic programming algorithm that builds a matrix of edit distances between substrings of the two strings.
* The matrix has dimensions (m+1) x (n+1), where m and n are the lengths of the two strings.
* We can initialize the matrix by setting the first row and column to be the edit distances between each string and the empty string. For example:



The edit distance between an empty string and any non-empty string is simply the length of the non-empty string, so the first row and column are just counting the number of characters in each string.

This relation says that if the characters at positions i-1 and j-1 in the two strings are the same, then the edit distance between those substrings is the same as the edit distance





6) Cosine Distance (from two paragraphs, find keywords, uska vector draw karna hai and then angle find krna hai)

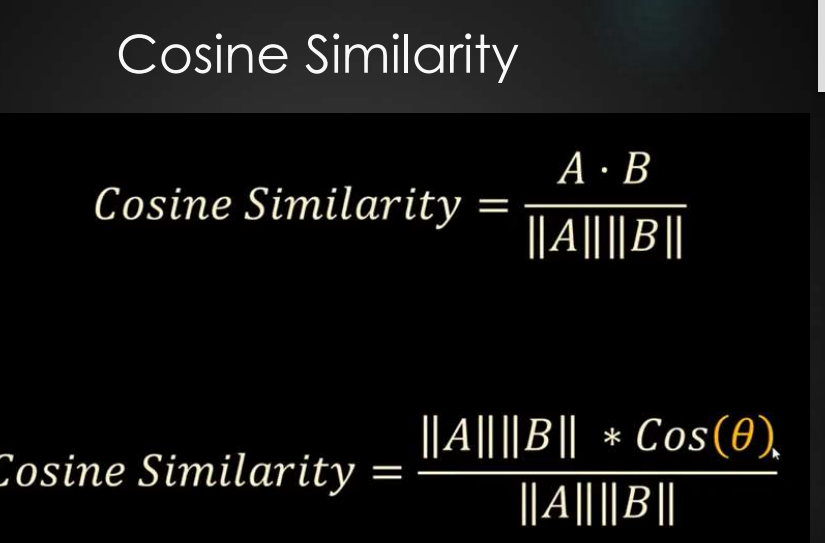
Text Similarity has to determine how the two text documents close to each other in terms of their context or meaning. There are various text similarity metric exist such as Cosine similarity, Euclidean distance and Jaccard Similarity. All these metrics have their own specification to measure the similarity between two queries.

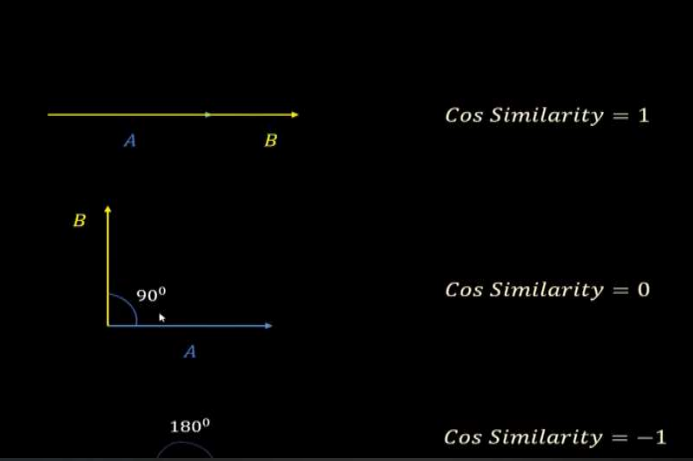
Cosine similarity is one of the metric to measure the text-similarity between two documents irrespective of their size in Natural language Processing. A word is represented into a vector form. The text documents are represented in n-dimensional vector space.

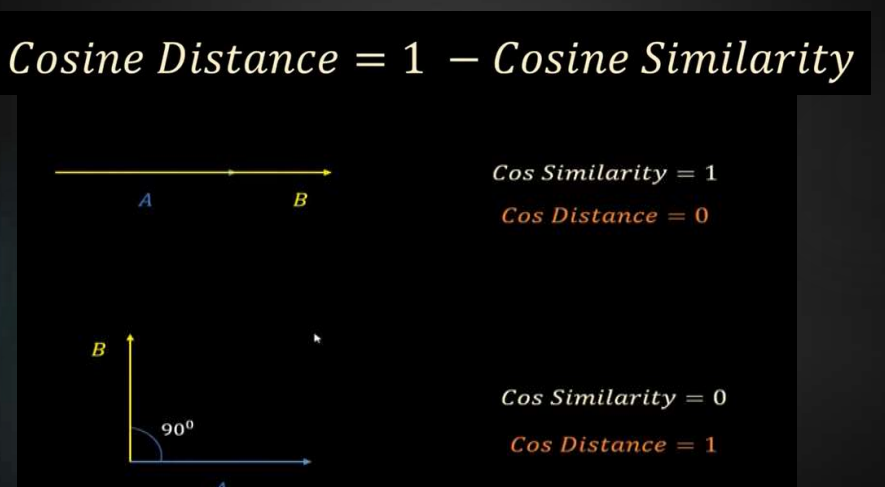
Mathematically, Cosine similarity metric measures the cosine of the angle between two n-dimensional vectors projected in a multi-dimensional space. The Cosine similarity of tw odocuments will range from 0 to 1. If the Cosine similarity score is 1, it means two vectors have the same orientation. The value closer to 0 indicates that the two documents have less similarity.

Cosine similarity is the cosine of the angle between two vectors and it is used as a distance evaluation metric between two points in the plane. The cosine similarity measure operates entirely on the cosine principles where with the increase in distance the similarity of data points reduces.

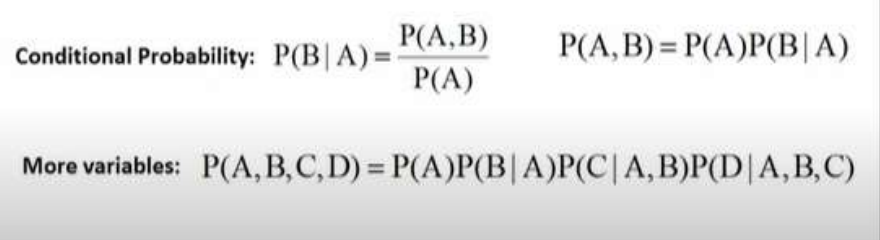
In Data Minning, similarity measure refers to distance with dimensions representing features of the data object, in a dataset. If this distance is less, there will be a high degree of similarity, but when the distance is large, there will be a low degree of similarity.

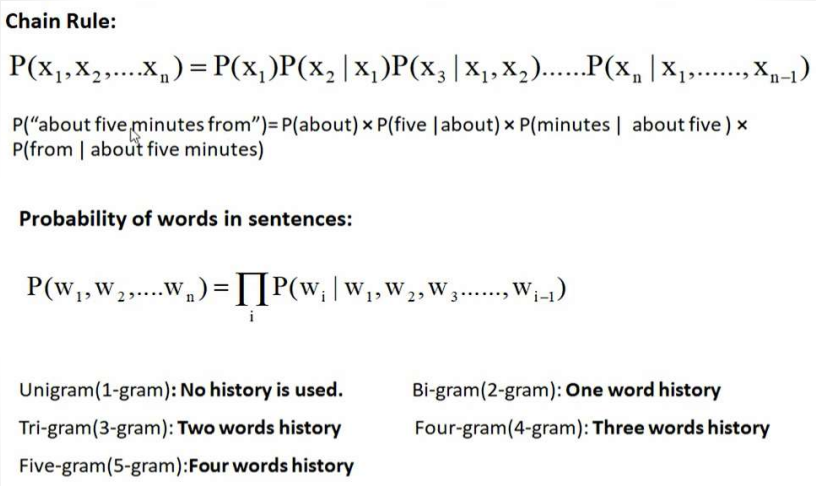


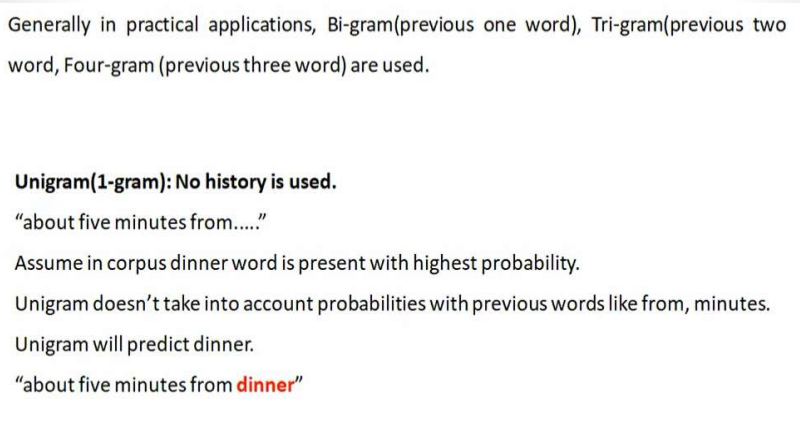


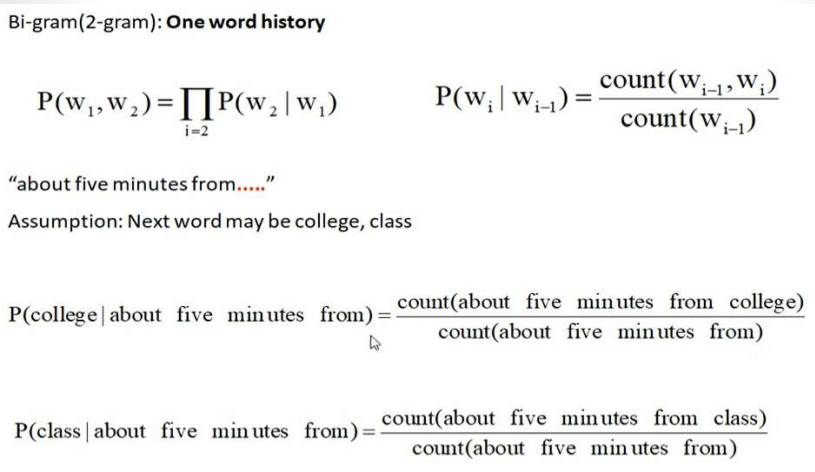


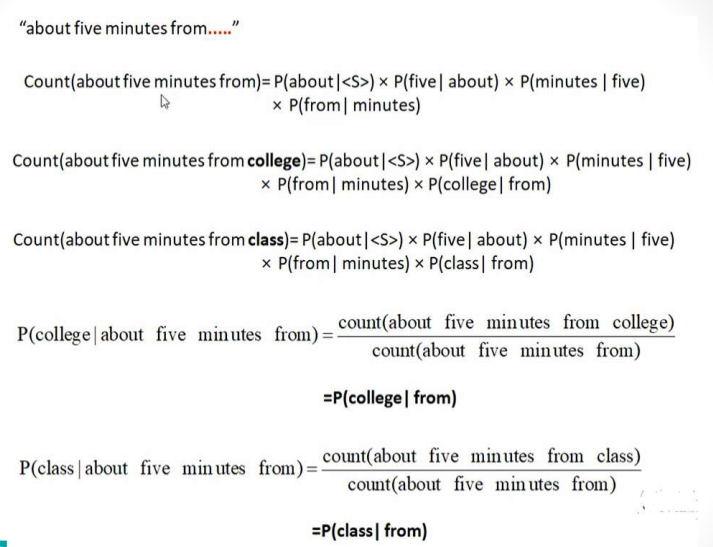
7) N-gram model(bigram, trigram, fourgram and how to overcome the zero probability by three methods)

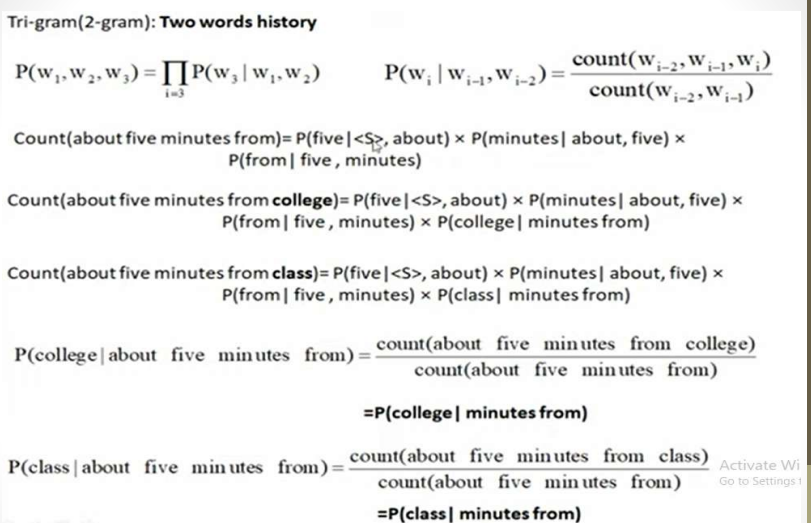
An n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.

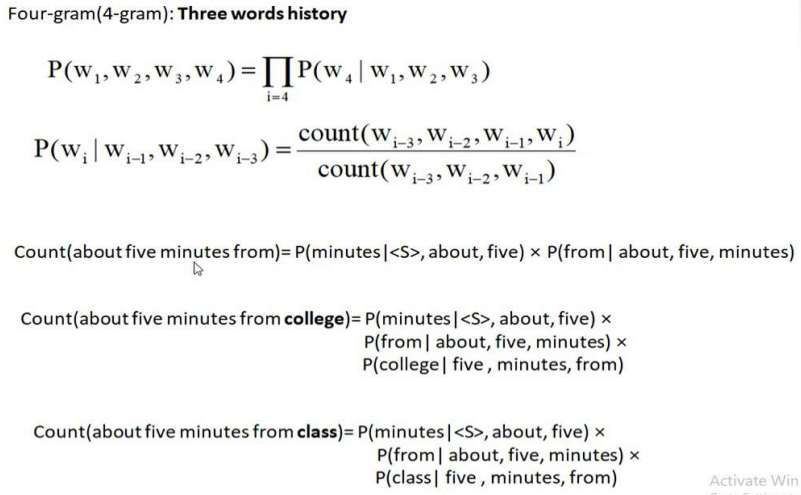


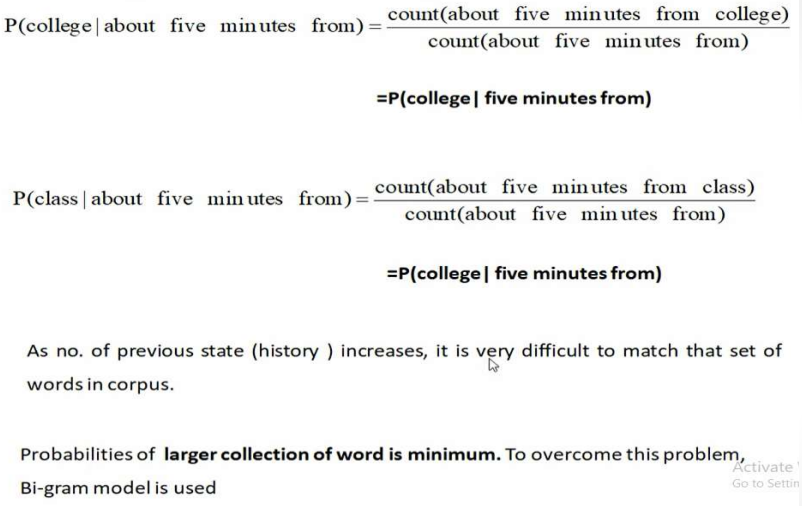




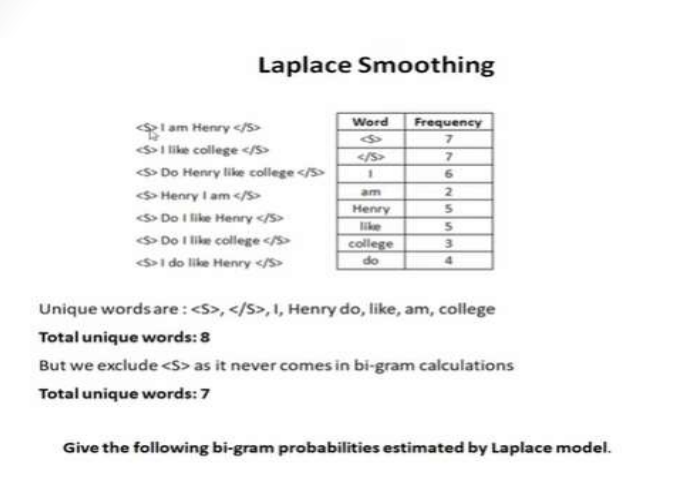


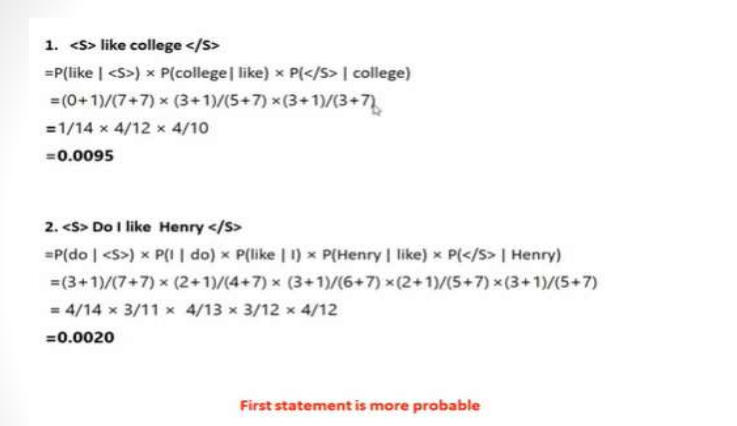


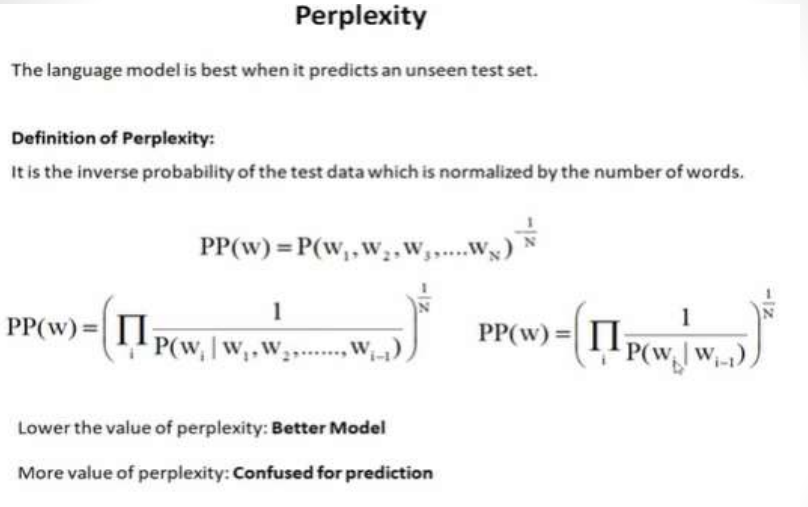




Laplace smoothing, also known as add-one smoothing, is a simple technique used to address the problem of zero probabilities in n-gram language models. In Laplace smoothing, a small constant (usually one) is added to the count of each unique n-gram and its corresponding context. This addition ensures that even unseen n-grams are assigned a non-zero probability.







8) word sense and word sense disambiguation

* A sense (or **word sense**) is a discrete representation of one aspect of the meaning of a word. Loosely following lexicographic tradition, we represent each sense with a superscript : bank1 and bank2, mouse1 and mouse2. In context, it's easy to see the different meanings :
  + mouse1 : .... a mouse controlling a computer system in 1968.
  + mouse2 : .... a quiet animal like a mouse
  + bank1 : ... a bank can hold the investments in a custodial account ...
  + bank2 :... as agriculture burgeons on the east bank, the river ...
* The senses of a word might not have any particular relation between them; it may be almost coincidental that they share an orthographic form. For example, the financial institution and sloping mound senses of bank seem relatively unrelated.
* In such cases we say that the two senses are homonyms, and the relation between the senses is one of homonymy. Thus bank1 ("financial institution") and bank2 ("sloping mound") are homonyms, as are the sense of bat meaning 'club for hitting a ball' and the one meaning 'nocturnal flying animal'. We say that these two uses of bank are homographs, as are the two uses of bat, because they are written the same.
* Two words can be homonyms in a different way if they are spelled differently but pronounced the same, like write and right, or piece and peace. We call these homophones; they are one cause of real-word spelling errors.

word sense disambiguation

What does this word mean?

* This plant needs to be watered each day.

= living plant

* This plant manufactures 1000 widgets each day.

= factory

We say that these words all have various word senses and that some of the senses are synonymous with one another. The process of choosing the right sense in context is called word sense disambiguation (or WSD).

WSD algorithms take as input a word in context and a fixed inventory of potential word senses and outputs the correct word sense in context.

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

**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**

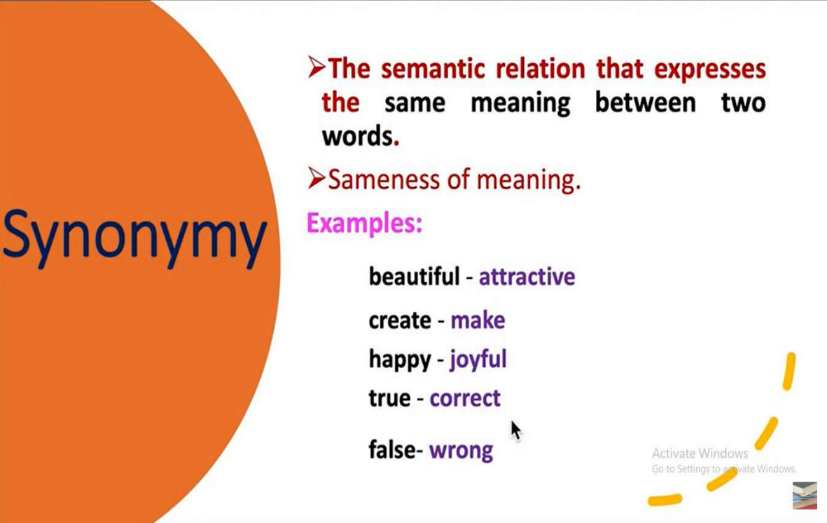
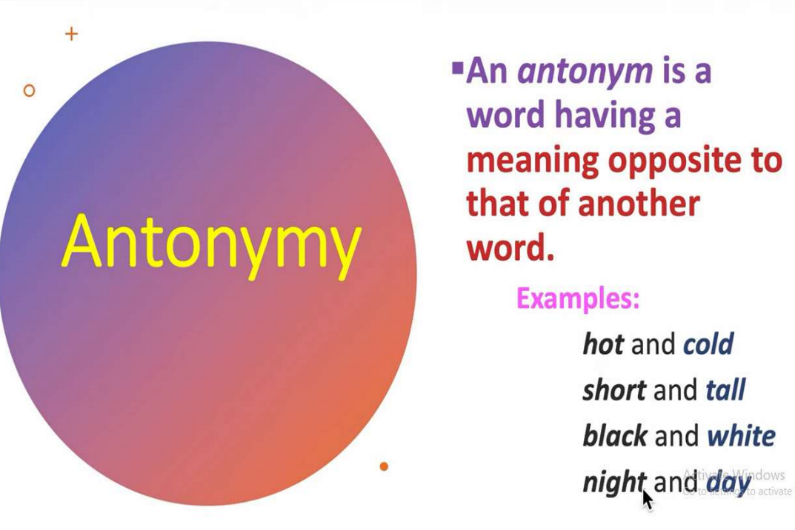
For disambiguation, machine learning methods make use of sense-annotated corpora to train. These methods assume that the context can provide enough evidence on its own to disambiguate the sense. In these methods, the words knowledge and reasoning are deemed unnecessary. The context is represented as 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**

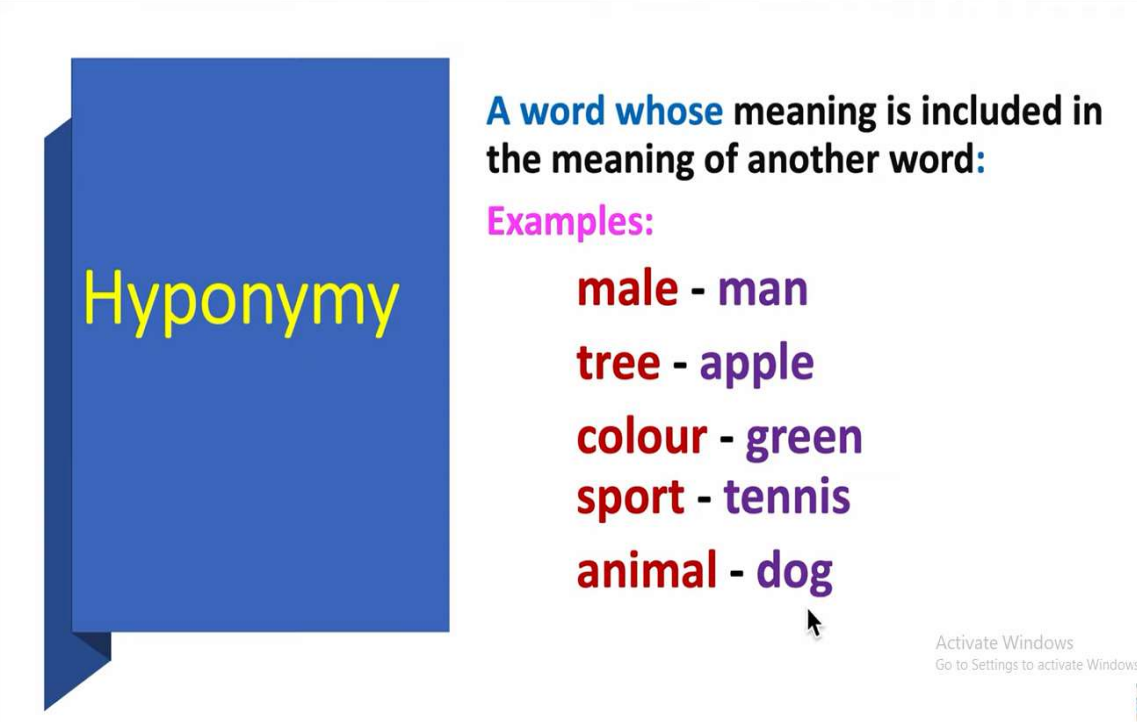
Due to the lack of training corpus, most of the word sense disambiguation algorithms use semi-supervised learning methods. It is because semi-supervised methods use both labelled as well as unlabeled data. These methods require very small amount of annotated text and large amount of plain unannotated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

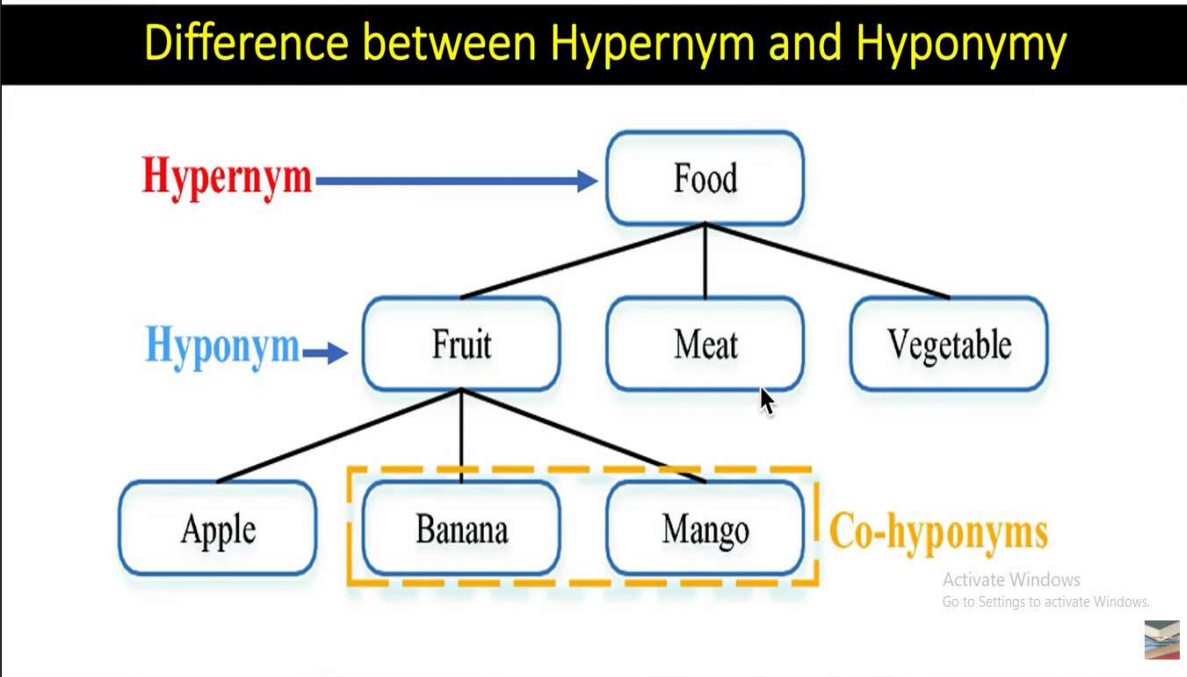
**Unsupervised Methods**

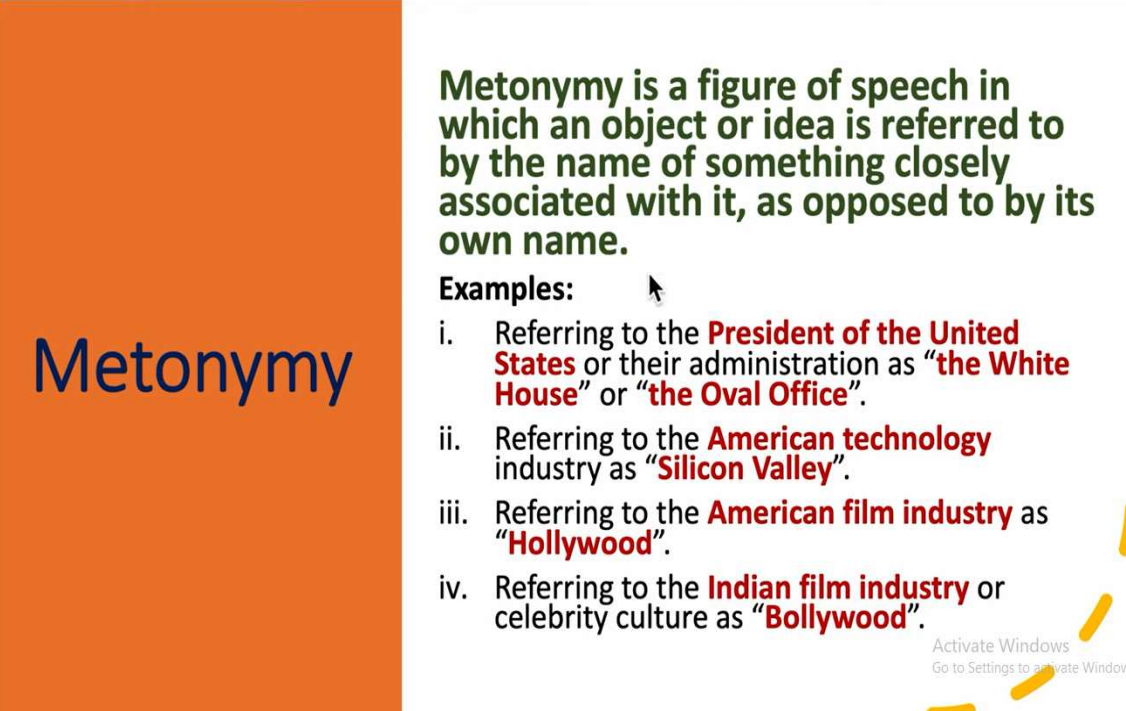
These methods assume that similar senses occur in similar context. That is why the senses can be induced from text by clustering word occurrences by using some measure of similarity of the context. This task is called word sense induction or discrimination. Unsupervised methods have great potential to overcome the knowledge acquisition bottleneck due to non-dependency on manual efforts.

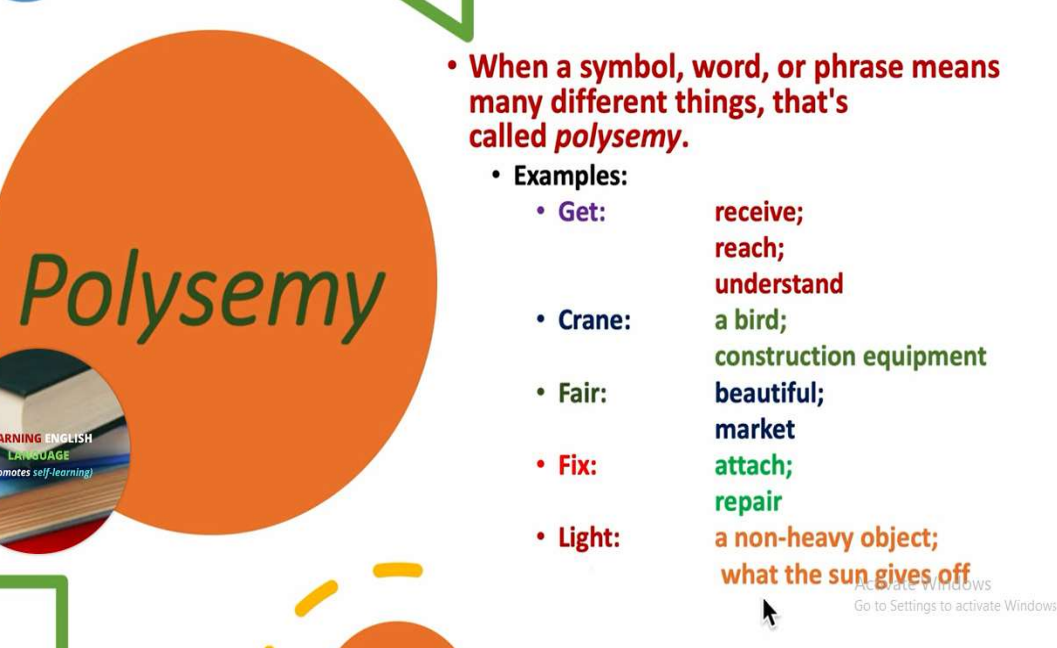
9) relation between synonym, antonym, hypernemy, hyponemy, polysemy.











10) word net

* WordNet is a lexical database of the English language.
* It groups words into sets of synonyms called synsets, provides short definitions, and records the various semantic relationships between these synonym sets.
* It's a resource commonly used in natural language processing and computational linguistics tasks, as well as in educational and research contexts.
* Developed at Princeton University in the Cognitive Science Laboratory starting in 1985, WordNet has become a fundamental resource in natural language processing (NLP), computational linguistics, and related fields.

**Key components and features of WordNet**

* **Synsets:** A synset, or synonym set, is a collection of words that are synonymous or closely related in meaning. Each synset represents a distinct concept or idea. For example, the synset for the word "dog" might include synonyms like "canine," "pooch," and "hound."
* **Words and Lemmas:** WordNet includes individual words, or lemmas, which are organized into synsets. Lemmas represent the base or dictionary form of a word. For example, the lemma "running" might be included in synsets related to activities or physical exercise.
* **Definitions:** Each synset in WordNet is accompanied by a short definition that describes the meaning of the concept represented by that synset. These definitions help users understand the nuances of each word's usage and meaning
* **Part-of-Speech (POS) Tags:** WordNet assigns part-of-speech tags to each word or lemma, indicating whether it functions as a noun, verb, adjective, or adverb.

This information is crucial for disambiguating word meanings and identifying appropriate synonyms in different contexts.

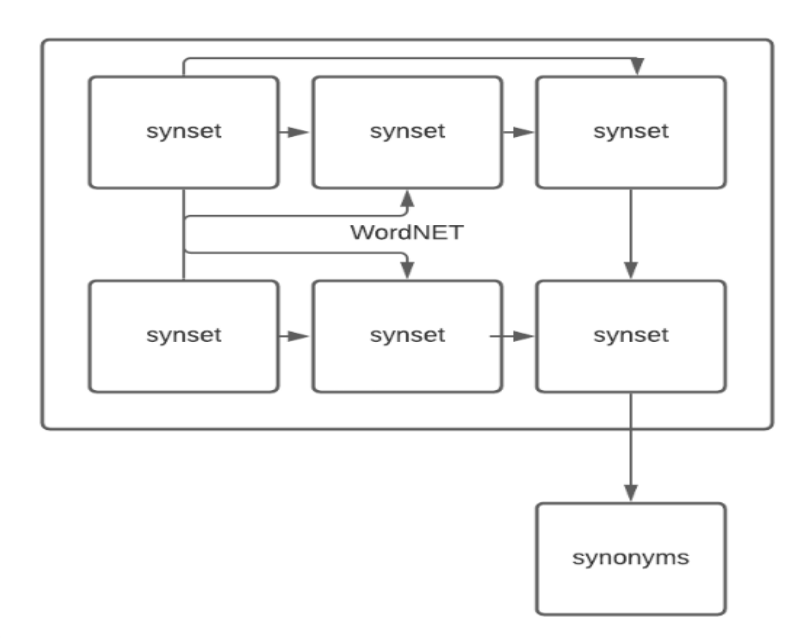
* **Semantic Relationships:** WordNet records various semantic relationships between synsets, providing valuable information about the connections between words and concepts. Some common semantic relationships in WordNet include:

• Hypernyms and Hyponyms: Hypernyms represent more general concepts, while hyponyms are more specific. For example, "animal" is a hypernym of "dog," and "dog" is a hyponym of "animal."

• Meronyms and Holonyms: Meronyms are parts or components of a whole, while holonyms are the wholes that contain those parts. For example, "wheel" is a meronym of "car," and "car" is a holonym of "wheel."

• Synonyms and Antonyms: WordNet identifies synonyms (words with similar meanings) and antonyms (words with opposite meanings) within and between synsets

* **Hierarchical Structure:** WordNet's synsets and semantic relationships are organized in a hierarchical structure, allowing users to navigate through different levels of abstraction and specificity. This hierarchical organization aids in understanding the relationships between concepts and facilitates various NLP tasks such as word sense disambiguation and semantic similarity measurement.
* **Applications:** WordNet is widely used in natural language processing tasks such as text analysis, information retrieval, machine translation, sentiment analysis, and more. Its rich collection of words, synsets, and semantic relationships provides a valuable resource for understanding and processing natural language text.



11) Categorization, summarisation, information retrieval, machine translation, sentiment analysis, named entity recognition(any 2)

* **Sentiment analysis** is a classification task in the area of natural language processing. Sometimes called 'opinion mining,' sentiment analysis models transform the opinions found in written language or speech data into actionable insights.
* Sentiment analysis involves determining whether the author or speaker's feelings are positive, neutral, or negative about a given topic. For instance, you would like to gain a deeper insight into customer sentiment, so you begin looking at customer feedback under purchased products or comments under your company's post on any social media platform. You would like to know if the customer is pleased with your services, neutral, or if he/she has any complaints, meaning whether the customer has a neutral, positive or negative sentiment regarding your products, services or actions. Figuring this out is called sentiment analysis.

**NLP methods for sentiment analysis**

* In classical methods, we define features and models that can then be identified by the sentiment analysis system. This can be done by :

1. Using a dictionary of manually defined keywords,
2. Creating a 'bag of words',
3. Using the TF-IDF strategy.

* Using a dictionary of manually defined keywords is based on the assumption that we know what words are typically associated with positive and negative emotions. For example, if we are going to classify movie reviews, we expect to find words such as "great", "super", and "love" in positive comments and words like "hate", "bad", and "awful" in negative comments.
* We can count the number of occurrences of every selected word to define feature vectors. Then, we can train a sentiment analysis classifier on each comment.
* This approach restricts you to manually defined words, and it is unlikely that every possible word for each sentiment will be thought of and added to the dictionary.
* This is where a **bag of words** comes in. Instead of calculating only words selected by domain experts, we can calculate the occurrences of every word that we have in our language (or every word that occurs at least once in all of our data). This will cause our vectors to be much longer, but we can be sure that we will not miss any word that is important for prediction of sentiment.
* We can think about **TF-IDF** as a modified version of the bag of words. Instead of treating every word equally, we normalize the number of occurrences of specific words by the number of its occurrences in our whole data set and the number of words in our document (comments, reviews, etc.). This means that our model will be less sensitive to occurrences of common words like "and", "or", "the", "opinion" etc., and focus on the words that are valuable for analysis.
* **Machine Translation** or MT or robotized interpretation is simply a procedure when computer software translates text from one language to another without human contribution. At its fundamental level, machine translation performs a straightforward replacement of atomic words in a single characteristic language for words in another.
* In simple language, we can say that machine translation works by using computer software to translate the text from one source language to another target language.

**Types of machine translation**

There are different types of machine translation.

1. **Statistical Machine Translation or SMT**

* It works by alluding to statistical models that depend on the investigation of huge volumes of bilingual content. It expects to decide the correspondence between a word from the source language and a word from the objective language. A genuine illustration of this is Google Translate.
* Presently, SMT is extraordinary for basic translation, however its most noteworthy disadvantage is that it doesn't factor in context, which implies translation can regularly be wrong or you can say, don't expect great quality translation.
* There are several types of statistical-based machine translation models which are: Hierarchical phrase-based translation, Syntax-based translation, Phrase-based translation, Word-based translation.

1. **Rule-based Machine Translation or RBMT**

* RBMT basically translates the basics of grammatical rules. It directs a grammatical examination of the source language and the objective language to create the translated sentence. But, RBMT requires broad editing, and its substantial reliance on dictionaries implies that proficiency is accomplished after a significant period.

1. **Hybrid Machine Translation or HMT**

* HMT, as the term demonstrates, is a mix of RBMT and SMT. It uses a translation memory, making it unquestionably more successful regarding quality. Nevertheless, even HMT has a lot of downsides, the biggest of which is the requirement for enormous editing, and human translators will also be needed. There are several approaches to HMT like multi-engine, statistical rule generation, multi-pass, and confidence-based.

1. **Neural Machine Translation or NMT**

* NMT is a type of machine translation that relies upon neural network models (based on the human brain) to build statistical models with the end goal of translation. The essential advantage of NMT is that it gives a solitary system that can be prepared to unravel the source and target text. Subsequently, it doesn't rely upon specific systems that are regular to other machine translation systems, particularly SMT.