



give me minimum edit distance algorithm in simple words to write in my sem end exams

-----



The minimum edit distance algorithm is a way to find out how similar two strings (like words or sentences) are by counting the minimum number of operations required to transform one string into the other. The operations typically include inserting a character, deleting a character, or substituting one character for another.

Here's a simple explanation of the algorithm, suitable for writing in an exam:

1. **Start with Two Strings:** Consider two strings for which you want to find the minimum edit distance, let's call them String A and String B.
2. **Create a Grid:** Make a grid (or table) where the rows represent the characters of String A and the columns represent the characters of String B. Include an extra row and column for the empty string at the beginning.
3. **Initialize the Grid:** Fill in the first row and the first column with numbers from 0 upwards. This represents the number of operations needed to transform an empty string to the current string.
4. **Fill the Grid:** Go through each cell in the grid, row by row and column by column. For each cell, calculate the minimum of three possibilities:
  - The value above plus 1 (representing a deletion).
  - The value to the left plus 1 (representing an insertion).
  - The value diagonally above and to the left plus 0 (if the corresponding characters in String A and B are the same) or plus 1 (if they are different, representing a substitution).
5. **Find the Minimum Edit Distance:** Once the grid is completely filled in, the value in the bottom-right cell will be the minimum edit distance between String A and String B.
6. **Traceback for Operations (Optional):** If you also need to find out which operations were performed, you can trace back from the bottom-right cell to the top-left cell, following the path of minimum values.

This algorithm is often used in computer science for tasks such as spell checking, plagiarism detection, and in bioinformatics for DNA sequencing and comparison.

-----



Write a RE to find the English article “the”. 1 CO1 K2 b What is meant by Word Normalization? 1 CO1 K1 c Write a formula to calculate Perplexity?

-----

a. **Regular Expression (RE) to Find the English Article "the"**: To create a regular expression that finds the English article "the", you can use `\bthe\b`. Here, `\b` is a word boundary in regular expressions, ensuring that "the" is recognized as a separate word and not as a part of another word (like "there" or "theme").

b. **Word Normalization**: Word normalization is the process in computational linguistics of transforming words into a standard or canonical form. This is often done to ensure that variations of a word are processed as the same item. Common methods of normalization include:

- Converting all letters to lower or upper case.
- Removing punctuations or special characters.
- Converting words to their root forms using stemming (cutting off prefixes/suffixes) or lemmatization (using vocabulary and morphological analysis).

c. **Formula to Calculate Perplexity**: The perplexity of a language model is a measurement of how well the model predicts a sample. It is calculated using the formula:

$$\text{Perplexity}(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} \quad \text{Perplexity}(W) = P(w_1, w_2, \dots, w_N)^{-N}$$

where:

- $W = w_1, w_2, \dots, w_N$   $W = w_1, w_2, \dots, w_N$  is a sequence of  $N$  words.
- $P(w_1, w_2, \dots, w_N)$   $P(w_1, w_2, \dots, w_N)$  is the probability of the word sequence according to the language model.
- $N$  is the number of words in the sequence.

In simpler terms, the perplexity is the inverse probability of the test set, normalized by the number of words. Lower perplexity indicates a better model that can more accurately predict the probability of a sequence of words.



Parse Tree for "I prefer a morning flight":

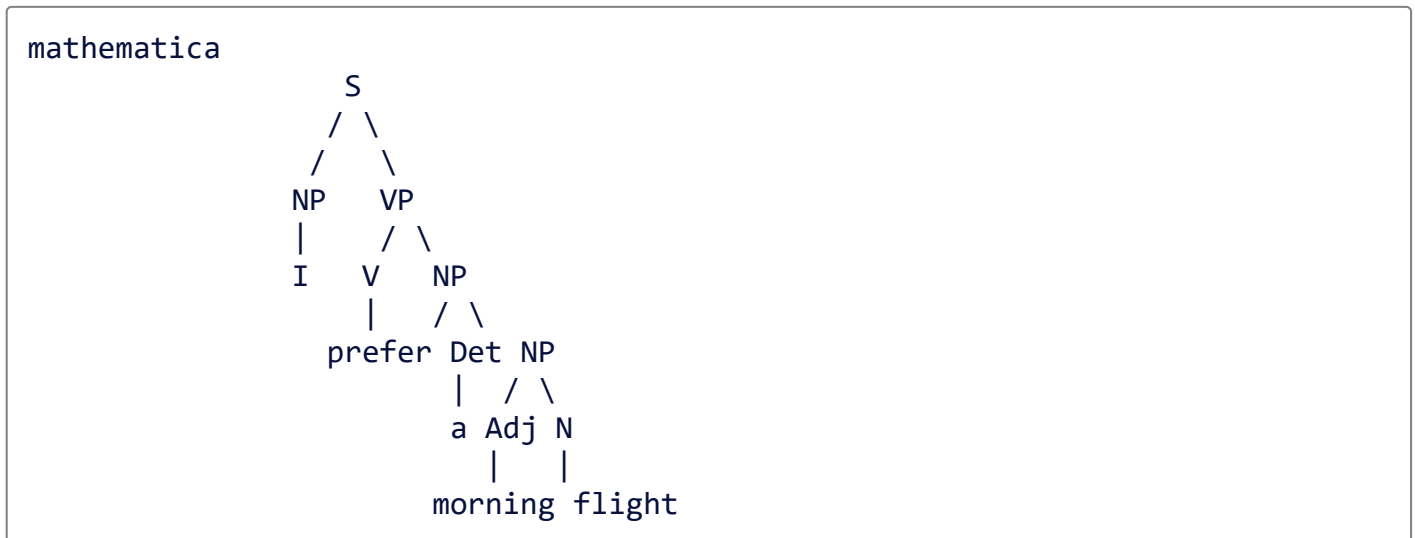
-----

Creating a parse tree for the sentence "I prefer a morning flight" involves breaking down the sentence into its grammatical components and arranging them hierarchically. Here's a basic breakdown of the tree structure:

1. **Sentence (S)**: The entire sentence forms the root of the tree.
  - **Noun Phrase (NP)**: This represents the subject of the sentence.
    - "I"
  - **Verb Phrase (VP)**: This contains the verb and the object of the sentence.
    - **Verb (V)**:

- "prefer"
- **Noun Phrase (NP):** This is the object of the verb.
  - **Determiner (Det):**
    - "a"
  - **Noun Phrase (NP):** Further description of the object.
    - **Adjective (Adj):**
      - "morning"
    - **Noun (N):**
      - "flight"

So, the structure of the parse tree can be visualized as follows:



This tree represents the syntactic structure of the sentence, showing how "I" is the subject, "prefer" is the verb, and "a morning flight" is the object of the sentence, with "a" as the determiner, and "morning" functioning as an adjective modifying the noun "flight".

-----



You:

Given the following short movie reviews, each labeled with a genre, either comedy or action: 1. fun, couple, love, love comedy 2. fast, furious, shoot action 3. couple, fly, fast, fun, fun comedy 4. furious, shoot, shoot, fun action 5. fly, fast, shoot, love action and a new document D: fast, couple, shoot, fly compute the most likely class for D.

-----



ChatGPT:

To determine the most likely class (genre) for the new document D, we can use a simple approach based on the frequency of words associated with each genre in the given short movie reviews. This approach is somewhat akin to the Naive Bayes classifier in text classification, though much simplified.

Here's how we can proceed:

1. Count the frequency of each word in the comedy and action categories from the given reviews.

2. Assign a score to document D for both genres based on the frequency of words in it that are associated with each genre.
3. The genre with the highest score will be the most likely class for document D.

Let's start by counting the frequencies:

### **Comedy Reviews:**

- fun: 3 times
- couple: 2 times
- love: 3 times

### **Action Reviews:**

- fast: 2 times
- furious: 2 times
- shoot: 3 times
- fly: 1 time

Now, let's score document D ("fast, couple, shoot, fly") for each genre:

### **Comedy Score:**

- fast: 0 (not in comedy)
- couple: 2
- shoot: 0 (not in comedy)
- fly: 0 (not in comedy)
- Total Comedy Score = 2

### **Action Score:**

- fast: 2
- couple: 0 (not in action)
- shoot: 3
- fly: 1
- Total Action Score = 6

Based on these scores, document D is more likely to belong to the **Action** genre, with a score of 6 compared to 2 for Comedy.

-----