**NLP NOTES TEE**

Tokenization and stemming are two important concepts in Natural Language Processing (NLP) that are often used in text processing and analysis.

1. **Tokenization:**
   * **Definition:** Tokenization is the process of breaking down a text into individual units, which can be words or subwords. These units are called tokens.
   * **Purpose:** The primary purpose of tokenization is to simplify the text and make it easier to analyze. It's a crucial step in NLP because many algorithms and models rely on individual words or subwords as basic units of input.
   * **Example:** Consider the sentence "Tokenization is an important step in NLP." The tokenized version of this sentence would be: ["Tokenization", "is", "an", "important", "step", "in", "NLP"].
2. **Stemming:**
   * **Definition:** Stemming is the process of reducing a word to its base or root form. The idea is to remove prefixes or suffixes from words, leaving behind the stem. This is done to group words with similar meanings.
   * **Purpose:** Stemming helps in reducing the dimensionality of the feature space and can improve the performance of text-based applications by treating different forms of the same word as a single entity.
   * **Example:** The stem of the words "running," "runner," and "ran" is "run."

**Types of Tokenization:**

1. **Word Tokenization:**
   * Breaks the text into words based on spaces or punctuation marks.
   * Example: Input sentence "Tokenization is important!" would be tokenized into ["Tokenization", "is", "important", "!"].
2. **Sentence Tokenization:**
   * Involves breaking down the text into sentences.
   * Example: Input paragraph with multiple sentences would be tokenized into a list of sentences.
3. **Subword Tokenization:**
   * Splits words into smaller subword units, useful for handling rare or out-of-vocabulary words.
   * Example: "unbelievable" might be tokenized into ["un", "believe", "able"].
4. **Character Tokenization:**
   * Tokenizes text into individual characters.
   * Example: "Tokenization" would be tokenized into ["T", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n"].

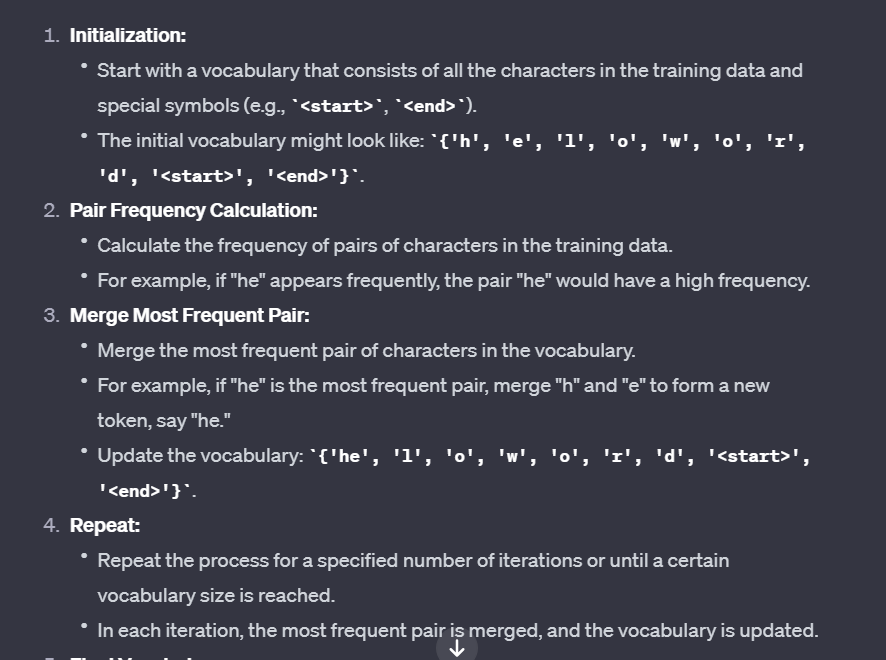
There are various tokenization algorithms and tools used in Natural Language Processing (NLP) to break down text into individual tokens. Here are some commonly used tokenization approaches:

1. **Whitespace Tokenization:**
   * **Description:** This simple method tokenizes text based on whitespace (spaces, tabs, or line breaks).
   * **Example:** Input sentence "Tokenization is important!" would be tokenized into ["Tokenization", "is", "important!"].
2. **NLTK Tokenization:**
   * **Description:** NLTK (Natural Language Toolkit) is a popular library in Python for NLP, and it provides various tokenization methods.
   * **Example:** NLTK's word\_tokenize function breaks text into words.
3. **Spacy Tokenization:**
   * **Description:** SpaCy is another widely used NLP library in Python, and it provides efficient tokenization along with other linguistic annotations.
   * **Example:** SpaCy's tokenization breaks text into words and includes information about part-of-speech and other linguistic features.
4. **Regex Tokenization:**
   * **Description:** Tokenization using regular expressions allows for more flexibility in defining tokenization rules.
   * **Example:** A regex pattern might be used to split text based on spaces, punctuation, or other criteria.
5. **SentencePiece:**
   * **Description:** SentencePiece is a unsupervised text tokenizer and detokenizer mainly designed for Neural Network-based text generation systems.
   * **Example:** It can be used for subword tokenization, breaking down words into smaller units.
6. **Byte Pair Encoding (BPE):**
   * **Description:** BPE is a subword tokenization algorithm that iteratively merges frequent pairs of characters or subwords.
   * **Example:** It can represent rare or out-of-vocabulary words by combining more common subwords.
7. **Tokenizer in Transformers Library (Hugging Face):**
   * **Description:** The Transformers library by Hugging Face includes tokenizers for various pre-trained language models like BERT, GPT, etc.
   * **Example:** The tokenizers in this library are often used for tokenizing input text compatible with specific pre-trained models.

**Types of Stemming:**

1. **Porter Stemming:**
   * Developed by Martin Porter, it's a widely used stemming algorithm.
   * It applies a series of rules to strip off common suffixes.
   * Example: "running," "runner," and "ran" would all be stemmed to "run."
2. **Lancaster Stemming:**
   * More aggressive than Porter stemming, as it applies a more extensive set of rules.
   * It can lead to more aggressive stemming, sometimes resulting in stems that are not valid words.
   * Example: "running" might be stemmed to "run" more aggressively than with Porter stemming.
3. **Snowball Stemming (Porter2):**
   * An improvement over the original Porter stemming algorithm.
   * More accurate in some cases, as it addresses some of the issues present in the Porter algorithm.
   * Example: Similar to Porter stemming, it aims to reduce words to their base or root form.
4. **Lovins Stemming:**
   * An algorithm that focuses on controlled stemming to produce morphological stems.
   * It considers linguistic aspects for stemming.
   * Example: "happiness" might be stemmed to "happy."

BytePair Encoding:



Lemmatization is a linguistic and natural language processing (NLP) technique that involves reducing words to their base or root form, known as the lemma. The lemma is the canonical, dictionary form of a word, which represents its meaning. Lemmatization helps in reducing inflected words to a common base or root, making it easier to analyze and compare words with similar meanings.

Here are some key points about lemmatization:

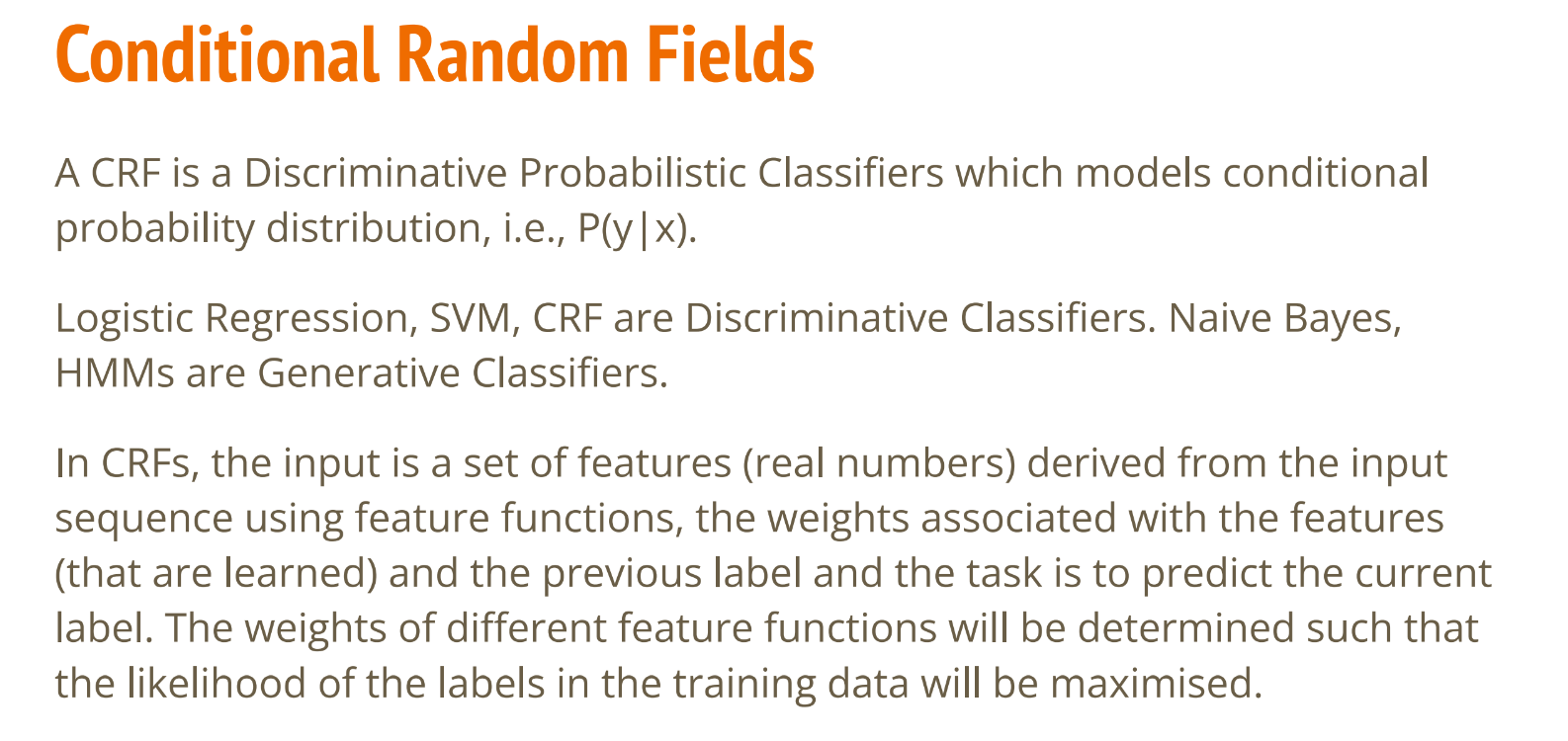
1. **Lemma vs. Inflected Form:**
   * The inflected form of a word includes variations like tense, number, gender, and case. For example, the word "run" may appear as "running," "ran," or "runs." The lemma for all these forms is "run."
2. **Purpose:**
   * Lemmatization is performed to normalize words, ensuring that different inflected forms of a word are treated as a single item.
   * It aids in improving the efficiency and effectiveness of text analysis by reducing the dimensionality of the feature space.
3. **Example:**
   * Consider the word "better." The lemma for "better" is "good." So, lemmatization would transform "better" to "good."
4. **Part-of-Speech (POS) Consideration:**
   * Lemmatization often takes into account the part-of-speech (POS) of a word. The lemma of a word can vary based on its POS. For instance, the lemma of "better" as an adjective is "good," while the lemma of "better" as a verb is "better."

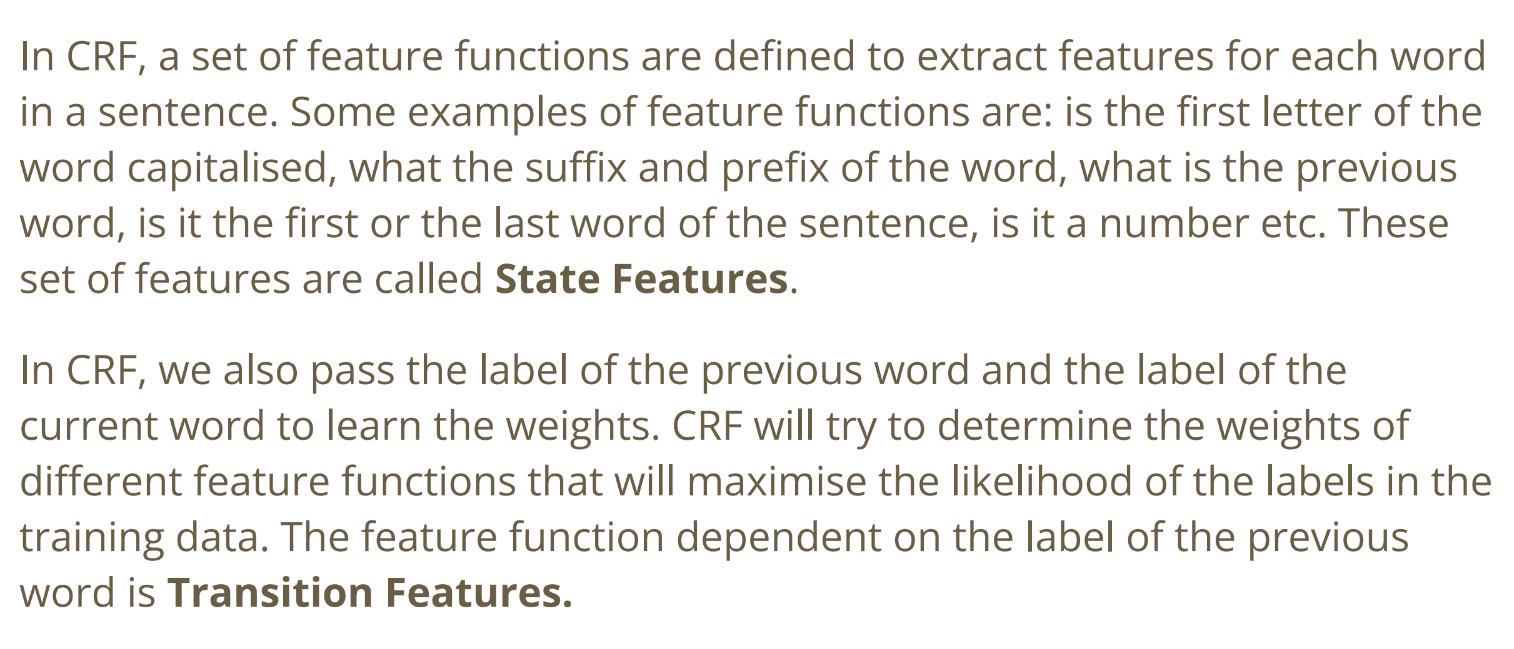
Lemma vs stem”

Lemmatization and stemming are both techniques used in natural language processing (NLP) to reduce words to their base or root form. However, there are key differences between the two approaches:

1. **Definition:**
   * **Stemming:** Involves removing prefixes or suffixes from words to obtain their root form. Stemming may result in the root form being a valid or invalid word.
   * **Lemmatization:** Involves reducing words to their base or dictionary form, known as the lemma. Lemmatization ensures that the resulting form is a valid word and represents the canonical, dictionary meaning.
2. **Output:**
   * **Stemming:** The output (stem) may be a partial word or a word that may not exist in the language.
   * **Lemmatization:** The output (lemma) is a valid word that exists in the language.
3. **Granularity:**
   * **Stemming:** Tends to be more aggressive, focusing on removing common prefixes and suffixes to achieve a common stem.
   * **Lemmatization:** Considers the broader linguistic context and aims to reduce words to their base form while preserving their meaning.
4. **Part-of-Speech (POS) Consideration:**
   * **Stemming:** Usually does not consider the part-of-speech of the word. It applies general rules to strip affixes.
   * **Lemmatization:** Often considers the part-of-speech of the word, ensuring that the lemma reflects the appropriate grammatical category.
5. **Algorithmic Approach:**
   * **Stemming:** Typically rule-based and involves applying a set of rules to truncate words.
   * **Lemmatization:** Can be rule-based or use machine learning models. It often relies on linguistic knowledge and dictionaries to map words to their lemmas.
6. **Use Cases:**
   * **Stemming:** Commonly used in information retrieval, search engines, and text mining tasks where the goal is to reduce words to a common root for pattern matching.
   * **Lemmatization:** Preferred in tasks where the underlying meaning of words is crucial, such as in question-answering systems, sentiment analysis, and machine translation.
7. **Examples:**
   * **Stemming:** The stem of "running" is "run," but the stem of "better" is "better."
   * **Lemmatization:** The lemma of "running" is "run," and the lemma of "better" is "good."

CRF for POS tagging:





**Query Expansion:** **https://github.com/ellisa1419/Wordnet-Query-Expansion/blob/master/wordnet.py**

**Definition:** Query expansion is a technique used in information retrieval and search engines to improve the relevance of search results by supplementing or expanding the original user query. The goal is to include additional terms or synonyms that might capture relevant information and improve the chances of retrieving the desired documents.

**Methods:**

* **Synonym Expansion:** Including synonyms of the original query terms to broaden the search.
* **Hypernym Expansion:** Adding broader, more general terms to the query.
* **Hyponym Expansion:** Including narrower, more specific terms.
* **Association-Based Expansion:** Adding terms that are statistically associated with the original query terms.

**Example:** If a user searches for "computer," query expansion might include terms like "desktop," "laptop," "hardware," and "software."

**Benefits:**

* Helps address issues with term mismatch and vocabulary variations.
* Expands the search scope, potentially retrieving documents that might have been missed with the original query.

**Word Sense Disambiguation (WSD):**

**Definition:** Word Sense Disambiguation is the process of determining the correct meaning or sense of a word when the word has multiple meanings or senses. Many words in natural language are ambiguous, and the correct interpretation often depends on the context in which the word is used.

**Challenges:**

* **Polysemy:** Words with multiple meanings.
* **Homonymy:** Words spelled the same but with different meanings.
* **Context Dependency:** The meaning of a word can depend on the surrounding words in a sentence.

**Methods:**

* **Supervised Learning:** Training a model on labeled examples to predict the correct sense.
* **Unsupervised Learning:** Clustering words based on their context to identify different senses.
* **Knowledge-Based Approaches:** Using external knowledge sources, such as dictionaries or ontologies.
* **Contextual Embeddings:** Leveraging pre-trained language models to capture contextual information.

**Example:** Consider the word "bat." Without context, it could refer to a flying mammal or a piece of sports equipment used in baseball.

**Applications:**

* **Machine Translation:** Ensuring accurate translation by understanding the intended sense of words.
* **Information Retrieval:** Improving search engine results by considering the correct sense of query terms.
* **Text Summarization:** Enhancing the accuracy of summarization by disambiguating ambiguous words.

In the context of Natural Language Processing (NLP), psycholinguistic theories are theories that draw on insights from psycholinguistics, the study of the psychological and cognitive factors involved in language processing. These theories aim to inform the design and understanding of computational models for processing natural language based on how humans perceive, produce, and comprehend language.

Key aspects of psycholinguistic theories in the context of NLP include:

1. **Word Recognition Models:**
   * Psycholinguistic theories often incorporate models of word recognition that emulate how humans process and recognize words in context. These models may take into account factors such as word frequency, contextual information, and semantic priming.
2. **Semantic Processing:**
   * Understanding the meaning of words and sentences is a crucial aspect of psycholinguistics. In NLP, psycholinguistic theories may influence the development of models that aim to capture the semantic relationships between words and phrases.
3. **Syntactic Processing:**
   * Psycholinguistic theories also consider how humans parse and understand the syntactic structure of sentences. Computational models in NLP may be inspired by these theories to improve syntactic parsing algorithms.
4. **Language Acquisition:**
   * Insights from psycholinguistics about how humans acquire language can inform the development of NLP systems, especially in the design of models for natural language understanding and generation.
5. **User Interaction and Interface Design:**
   * Psycholinguistic theories can be valuable in designing NLP interfaces that are more intuitive and user-friendly. Understanding how humans naturally process language can guide the development of conversational agents and chatbots.
6. **Error Analysis and Correction:**
   * Understanding how humans make errors in language processing can be useful in designing robust NLP systems that can handle noisy or imperfect input. Psycholinguistic theories may contribute to error analysis and correction strategies.
     + .
7. **Lexical Semantics:**
   * **Definition:** Lexical semantics is a subfield of linguistics and natural language processing (NLP) that focuses on the meaning of words and how they combine to form the meaning of phrases and sentences.
   * **Key Concepts:**
     + **Word Sense Disambiguation (WSD):** Resolving ambiguity in the meaning of words with multiple senses.
     + **Lexical Relations:** Understanding how words relate to each other, such as synonyms, antonyms, hypernyms, and hyponyms.
     + **Lexical Semantics in NLP:** In NLP, lexical semantics is crucial for tasks like sentiment analysis, information retrieval, and machine translation. Understanding the nuances of word meanings contributes to the accuracy of these applications.

SVD:

SVD stands for Singular Value Decomposition, and it is a mathematical technique used in linear algebra and numerical analysis. SVD is widely applied in various fields, including signal processing, statistics, machine learning, and natural language processing. It is a way of decomposing a matrix into three other matrices, providing a concise and efficient representation of the original matrix.

For a given matrix A of dimensions m x n, the SVD factorizes it into three matrices:

�=�Σ��*A*=*U*Σ*VT*

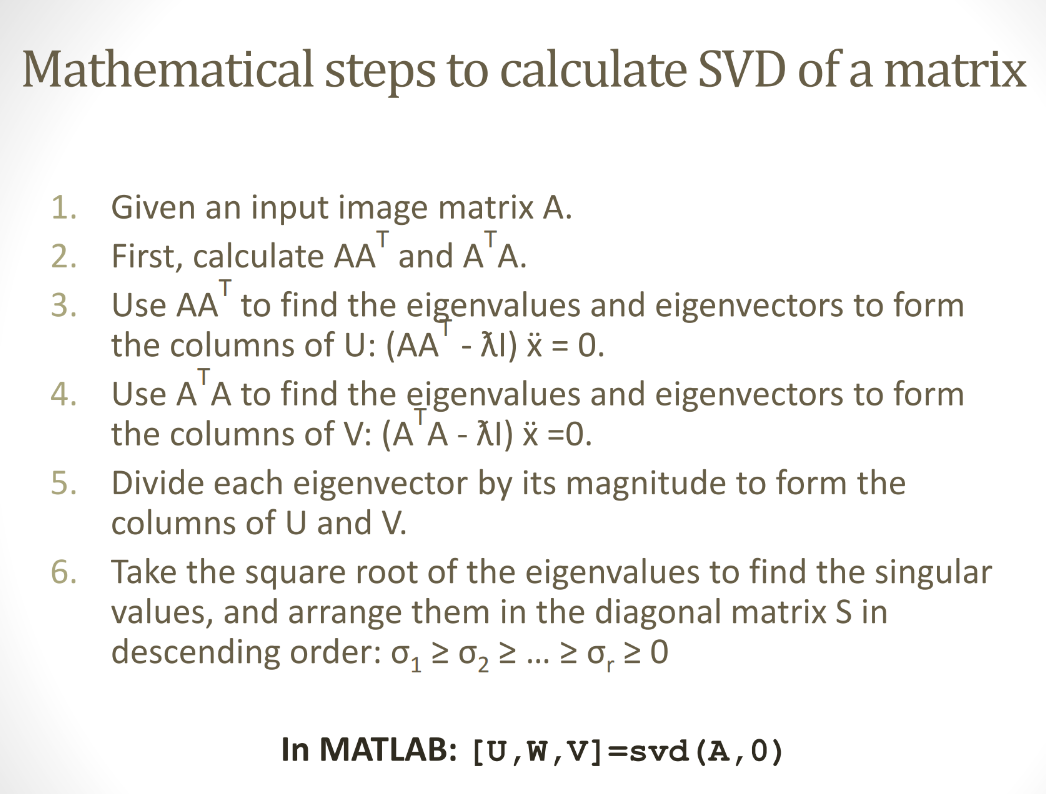
where:

* �*U* is an m x m orthogonal matrix (i.e., ���=���=�*UTU*=*UUT*=*I*, where I is the identity matrix).
* ΣΣ is an m x n diagonal matrix with non-negative real numbers on the diagonal. These non-negative numbers are the singular values of the original matrix A.
* ��*VT* is the transpose of an n x n orthogonal matrix V.

SVD is especially useful in various applications, including:

1. **Dimensionality Reduction:**
   * SVD can be used to reduce the dimensionality of a dataset by keeping only the most significant singular values and their corresponding columns in U and V.
2. **Matrix Approximation:**
   * SVD allows the approximation of a matrix by truncating the singular value decomposition, providing a low-rank approximation that retains the most important information.
3. **Solving Linear Systems:**
   * SVD can be applied to solve linear systems of equations and is particularly useful when the system is over-determined or when the matrix is ill-conditioned.
4. **Principal Component Analysis (PCA):**
   * SVD is a fundamental component of PCA, a technique used for dimensionality reduction and data analysis.
5. **Image Compression:**
   * SVD is employed in image compression by representing images as a sum of outer products of vectors, where each vector corresponds to a singular value and its associated columns in U and V.
6. **Latent Semantic Analysis (LSA) in NLP:**
   * SVD is used in natural language processing for latent semantic analysis, where it helps uncover the relationships between terms and documents in a term-document matrix.

In summary, Singular Value Decomposition is a versatile mathematical tool that finds applications in a range of fields, providing a powerful method for matrix decomposition and analysis.



RAKE AND YAKE:

RAKE (Rapid Automatic Keyword Extraction) and YAKE (Yet Another Keyword Extractor) are two algorithms designed for automatic keyword extraction from text. Both algorithms aim to identify and extract key terms or phrases that represent the main topics or themes within a given document or set of documents.

**RAKE (Rapid Automatic Keyword Extraction):**

**Key Features:**

1. **Candidate Generation:**
   * RAKE generates potential keywords by splitting the text into individual words and phrases (candidates) using punctuation and stop words as delimiters.
2. **Keyword Score Calculation:**
   * Scores are assigned to candidate keywords based on the co-occurrence of words within the text. The more often a word appears in proximity to others, the higher its score.
3. **Phrase Extraction:**
   * RAKE considers both single words and multi-word phrases as potential keywords. This allows it to capture meaningful phrases that might represent key topics.
4. **Scoring Algorithm:**
   * The score of a candidate keyword is calculated by dividing the degree (number of unique words in the candidate) by its frequency (number of occurrences). Candidates with higher scores are considered more important.

**Pros:**

* Simplicity and efficiency.
* No need for training data.

**Cons:**

* Limited in handling context or semantic relationships between words.

**YAKE (Yet Another Keyword Extractor):**

**Key Features:**

1. **Text Preprocessing:**
   * YAKE tokenizes the input text and removes stop words and other irrelevant terms.
2. **Term Extraction:**
   * YAKE uses a data-driven approach to identify terms and phrases. It considers frequent sequences of words as potential terms.
3. **Term Scoring:**
   * YAKE employs a term scoring method based on the relative frequency of terms in the given text and the distribution of these terms across the entire corpus.
4. **Keyword Selection:**
   * YAKE uses a threshold to select keywords based on their scores. Terms with scores above the threshold are considered keywords.
5. **Multi-lingual Support:**
   * YAKE is designed to work across multiple languages.

**Pros:**

* Considers term frequency and distribution.
* Multilingual support.

**Cons:**

* May require tuning of parameters like the threshold.

**Comparison:**

* Both RAKE and YAKE are unsupervised keyword extraction algorithms that don't require labeled training data.
* RAKE relies on co-occurrence and frequency, while YAKE considers both term frequency and distribution across the corpus.
* YAKE supports multiple languages, making it more versatile in a multilingual context.

In summary, RAKE and YAKE are algorithms for automatic keyword extraction, each with its own approach and strengths. The choice between them may depend on specific requirements and characteristics of the text data being processed.