- Semantic relationships between distributionally similar words
    - Antonymy – Words with opposite meaning “hot” “cold”
    - Synonymy – Words with similar meaning “big” “large”
    - Hyponymy/Hypernymy – “rose hyponym” “flower hyperpym”

- Zipfs law
    - Zipf's Law is an empirical statistical principle named after the linguist George Zipf, who observed it in the distribution of word frequencies in natural languages. It states that in a corpus of text, the frequency of any word is inversely proportional to its rank in the frequency table. In simpler terms, Zipf's Law suggests that the most frequent word occurs approximately twice as often as the second most frequent word, three times as often as the third most frequent word, and so on.

- Word2Vec and GloVe are both popular algorithms used for generating word embeddings, which are vector representations of words in a continuous vector space. These embeddings capture semantic relationships between words based on their contexts in large text corpora. While both algorithms serve a similar purpose, they have differences in their methodologies and characteristics. Here are the main similarities and differences between Word2Vec and GloVe:
- Similarities:
    - Word Embeddings: Both Word2Vec and GloVe generate word embeddings, which represent words as dense, continuous vectors in a high-dimensional space.
    - Unsupervised Learning: Both algorithms are trained in an unsupervised manner, meaning they learn from raw text data without explicit human supervision.
    - Semantic Relationships: They both capture semantic relationships between words. Words with similar meanings or contexts are represented by vectors that are closer together in the embedding space.
    - Efficiency: Both algorithms are relatively efficient and scalable, making them applicable to large text corpora.
- Differences:
    - Methodology:
    - Word2Vec: It uses shallow neural networks, specifically either the Continuous Bag of Words (CBOW) or Skip-gram architectures, to learn word embeddings by predicting surrounding words given a target word or vice versa.
    - GloVe: It leverages co-occurrence statistics of words in a corpus to directly learn word embeddings. It constructs a global word-word co-occurrence matrix from the corpus and then factorizes this matrix to obtain word embeddings.
    - Training Objective:
    - Word2Vec: It aims to maximize the probability of predicting context words given a target word (CBOW) or vice versa (Skip-gram).
    - GloVe: It aims to directly model the ratios of word-word co-occurrence probabilities. The objective is to minimize the difference between the dot

product of word embeddings and the logarithm of their co-occurrence probabilities.
    - Context Window:
    - Word2Vec: The context window size is a parameter that determines the number of surrounding words considered as context for predicting the target word.
    - GloVe: It typically considers the entire text corpus for computing co-occurrence statistics, rather than using a fixed-size context window.
    - Speed and Memory Usage:
    - Word2Vec: It tends to be faster to train compared to GloVe, especially for large datasets. It also consumes less memory during training.
    - GloVe: Training GloVe requires constructing and factorizing a large co-occurrence matrix, which can be memory-intensive and slower compared to Word2Vec, particularly for very large corpora.
- In summary, both Word2Vec and GloVe are effective at learning word embeddings and capturing semantic relationships in text data, but they differ in their underlying methodologies, training objectives, and computational characteristics. The choice between them depends on factors such as the size of the dataset, computational resources available, and specific requirements of the application.

- To compute the probability of a sentence using a trigram model:
    - Tokenize the sentence and add start/end tokens.
    - Calculate the conditional probability of each word given its two preceding words.
    - Multiply these probabilities to get the overall sentence probability.
    - Apply smoothing if needed to handle unseen trigrams.
    - Normalize the probabilities.
    - Compute the final probability of the sentence.

- Perplexity measures how well a language model predicts data. Lower values indicate better performance, with 1 being perfect. It's calculated from the entropy of the model's predictions on a test dataset.

- LSTMs (Long Short-Term Memory networks) have several advantages over vanilla RNNs (Recurrent Neural Networks) in language modeling:
    - Long-term Dependencies: LSTMs are better at capturing long-term dependencies in sequences compared to vanilla RNNs. This is because LSTMs have a more sophisticated memory mechanism that allows them to remember information over longer time steps without suffering from the vanishing gradient problem.
    - Gradient Flow: LSTMs address the vanishing gradient problem better than vanilla RNNs. They achieve this by using gated units and a carefully designed architecture that helps in preserving and flowing gradients through many time steps during training.
    - Explicit Memory Cells: LSTMs have explicit memory cells that can selectively remember or forget information over time. This enables them

to retain relevant information for longer periods and discard irrelevant or outdated information, making them more effective for tasks like language modeling.
    - Gating Mechanism: LSTMs utilize gating mechanisms (input gate, forget gate, and output gate) that regulate the flow of information through the cell. These gates control how much information is retained or discarded at each time step, allowing LSTMs to effectively learn and adapt to complex sequential patterns.
    - Less Susceptible to Exploding Gradients: While both vanilla RNNs and LSTMs can suffer from exploding gradients, LSTMs are generally less susceptible to this issue due to their gating mechanisms and better gradient flow. This makes them more stable and easier to train, especially on longer sequences.
- Overall, LSTMs are a more sophisticated and powerful variant of RNNs, particularly for tasks like language modeling where capturing long-term dependencies is crucial. Their ability to address the vanishing gradient problem, handle long sequences, and selectively retain important information makes them a preferred choice over vanilla RNNs in many applications.
- Combining a character-based network with a word-based network in language modelling can offer several advantages:
    - Handling Out-of-Vocabulary (OOV) Words: Character-based networks are effective at handling out-of-vocabulary words because they can generate embeddings for unseen words based on their character compositions. This helps improve coverage, especially for rare or unseen words that may not be present in the vocabulary of a word-based model.
    - Handling Morphologically Rich Languages: Languages with rich morphology often have complex word forms that may not be captured well by word-based models. Character-based models excel at capturing morphological variations by directly processing the characters of words, making them more suitable for languages with complex morphology.
    - Improving Subword Representation: By combining character-level representations with word-level representations, you can create more robust subword embeddings. This can help in capturing both fine-grained character-level information and coarse-grained word-level semantics, leading to richer and more comprehensive representations.
    - Addressing Misspellings and Typos: Character-based models are inherently more robust to misspellings and typos because they can still generate meaningful representations for words based on their character sequences. By incorporating character-level information, the combined model can better handle noisy or misspelled text.
    - Enhancing Rare Word Representations: Rare words with low frequency in the training data often suffer from poor representations in word-based models. By leveraging character-level information, the combined model can create more informative embeddings for rare words, improving their representation and the overall performance of the language model.
- To combine a character-based network with a word-based network, you can employ various architectures such as:

- Hybrid Models: Train separate character-based and word-based models and then combine their representations at a higher level, such as through concatenation or summation, before feeding them into subsequent layers.
- Multi-Input Models: Design neural network architectures that accept both character-level and word-level inputs simultaneously. This allows the model to learn representations from both levels of granularity in parallel.
- Attention Mechanisms: Incorporate attention mechanisms that dynamically weigh the importance of character-level and word-level information based on the context. This enables the model to selectively focus on relevant information from both levels during the language modeling task.
- By combining character-based and word-based networks, you can create more versatile and robust language models that effectively capture both fine-grained and coarse-grained linguistic features, leading to improved performance across various tasks and linguistic domains.

Conditional Random Fields (CRFs) are a type of probabilistic graphical model that's particularly well-suited for sequence labeling tasks. Here are some types of problems for which CRFs are typically used:

Named Entity Recognition (NER): Identifying and classifying entities such as names of persons, organizations, locations, dates, and numerical expressions in text documents.
Part-of-Speech (POS) Tagging: Assigning grammatical categories (e.g., noun, verb, adjective) to words in a sentence to analyze its syntactic structure.
Chunking: Identifying and labeling contiguous sequences of words or tokens, such as noun phrases or verb phrases, within sentences.
Semantic Role Labeling (SRL): Identifying the predicate-argument structure of a sentence by labeling the semantic roles of words or phrases, such as identifying the subject, object, or verb of a sentence.
Named Entity Recognition from Speech: Recognizing named entities in speech transcripts or transcribed spoken dialogues.
Biology and Bioinformatics: Analyzing biological sequences such as DNA, RNA, and protein sequences for tasks like gene finding, protein structure prediction, and sequence alignment.
Natural Language Understanding (NLU): Tasks related to understanding and extracting meaning from natural language text, such as sentiment analysis, information extraction, and question answering.
CRFs are advantageous for these problems because they model the dependencies between labels in a sequence, capturing contextual information that's important for accurate labeling. They can handle both local and global features, taking into account the entire sequence when making predictions. This makes CRFs especially useful for tasks where the context of each element in the sequence is crucial for accurate labelling

Condensed version of above

Here's a condensed version of your provided text:

CRFs Usage: CRFs excel in sequence labeling tasks such as Named Entity Recognition (NER), Part-of-Speech (POS) Tagging, Chunking, and Semantic Role Labeling (SRL). They're also valuable in biology, bioinformatics, and Natural Language Understanding (NLU).
Advantages of CRFs: CRFs model dependencies between labels in a sequence, enabling accurate labeling by capturing contextual information. They handle local and global features, considering the entire sequence for predictions.
Combining Networks: Combining character-based and word-based networks enhances language modeling, providing better coverage, handling complex languages, and improving representation for rare words. Architectures like hybrid models, multi-input models, or attention mechanisms facilitate this combination.
LSTMs vs. RNNs: LSTMs outperform vanilla RNNs in language modeling due to better handling of long-term dependencies, addressing vanishing gradient issues, explicit memory cells, gated mechanisms, and reduced susceptibility to exploding gradients.
Word2Vec vs. GloVe: Both algorithms generate word embeddings, but Word2Vec uses neural networks for prediction, while GloVe leverages co-occurrence statistics. Their training objectives, context windows, and speed differ, impacting their suitability for various applications.
Zipf's Law: Zipf's Law describes the distribution of word frequencies in a corpus, stating that the frequency of any word is inversely proportional to its rank in the frequency table. It's crucial for understanding language structures and patterns.
Trigram Model: Trigram models compute the probability of a sentence by considering the conditional probabilities of each word given its two preceding words. This involves tokenization, probability calculation, smoothing, normalization, and final probability computation.
Perplexity: Perplexity measures a language model's ability to predict data, with lower values indicating better performance. It's calculated from the entropy of the model's predictions on a test dataset

The difference between generative and discriminative statistical classification models lies in their approach to modeling the probability distribution of the data and their decision boundaries:

Generative Models:
Modeling Approach: Generative models learn the joint probability distribution P(X,Y) of the input features X and the class labels Y.
Probability Estimation: They estimate the likelihood of observing the input features given each class, P(Y).
Decision Making: To classify a new instance, generative models use Bayes' theorem to calculate the posterior probability of each class given the input features, P(Y|X). They then choose the class with the highest posterior probability.
Examples: Naive Bayes, Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs).
Discriminative Models:
Modeling Approach: Discriminative models directly learn the conditional probability distribution P(Y|X) of the class labels given the input features.
Probability Estimation: They focus on estimating the decision boundary that separates different classes in the feature space, without explicitly modeling the underlying data distribution.
Decision Making: To classify a new instance, discriminative models directly predict the class label based on the input features, without explicitly computing class-conditional probabilities.
Examples: Logistic Regression, Support Vector Machines (SVMs), Neural Networks (when used for classification).
Key Differences:

Focus: Generative models focus on modeling the entire data distribution and use it to infer class labels, while discriminative models focus solely on learning the decision boundary between classes.
Probability Estimation: Generative models explicitly estimate both class-conditional probabilities and prior probabilities, allowing for probabilistic inference. Discriminative models directly estimate the conditional probability of class labels given input features.
Decision Making: Generative models involve calculating posterior probabilities and selecting the class with the highest probability. Discriminative models make direct predictions based on the learned decision boundary.
Application Considerations:

Data Complexity: Generative models can handle missing data and can be more robust when the data distribution is complex. Discriminative models may perform better when the decision boundary is more important than the data distribution itself.
Model Interpretability: Generative models provide explicit probabilistic interpretations of the data, which can be beneficial for certain tasks. Discriminative models may offer simpler decision boundaries but may lack probabilistic interpretability.
Scalability: Discriminative models often have simpler parameterizations and can be more computationally efficient, making them suitable for large-scale datasets.

Generative models may be more computationally intensive due to the need to model the entire data distribution.

Generative Models:

- Hidden Markov Models (HMMs): Probabilistic models that represent a sequence of observable events (words) generated by a sequence of hidden states.
- Naïve Bayes Classifier: A simple probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence among features.
- Maximum Entropy Markov Model (MEMM): A discriminative probabilistic model that extends HMMs by incorporating features from the observed data.
- Conditional Random Field (CRF): A discriminative probabilistic model used for labeling sequential data, considering the dependencies between adjacent labels.

Discriminative Models:

- Logistic Regression: A linear discriminative model used for binary classification, which models the probability of the class label given the input features.
- Support Vector Machines (SVMs): Discriminative models that find the hyperplane that maximally separates different classes in the feature space.
- Random Forest Classifier: Ensemble learning methods that construct a multitude of decision trees during training and output the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- Gradient Boosting Classifier: A machine learning technique for regression and classification problems that produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Semantic Analysis (LSA), and Non-Negative Matrix Factorization (NNMF) all have the following commonalities:

1. **Dimensionality Reduction**: All of these techniques are used for reducing the dimensionality of the data. They aim to represent high-dimensional data in a lower-dimensional space while preserving as much relevant information as possible.
2. **Linear Algebra Basis**: They are all based on linear algebra techniques. PCA, SVD, and NNMF directly utilize linear algebra operations to decompose or factorize the data matrix, while LSA is a specific application of SVD in the context of text mining and natural language processing.
3. **Matrix Factorization**: PCA, SVD, LSA, and NNMF involve decomposing or factorizing a matrix into lower-dimensional representations or components. In PCA and LSA, the matrix is typically a covariance or similarity matrix, while in SVD and NNMF, it is the original data matrix.
4. **Orthogonal Transformation**: PCA, SVD, and NNMF involve orthogonal transformations or rotations to find the principal components, singular vectors,

or non-negative basis vectors that best capture the variance or structure of the data. These transformations ensure that the resulting components are orthogonal to each other.

5. **Dimension Reduction and Feature Extraction**: These techniques are commonly used for feature extraction and dimensionality reduction in various fields, including image processing, natural language processing, recommendation systems, and data analysis.
6. **Applications**: They find applications in a wide range of domains, including image compression, text mining, topic modeling, collaborative filtering, and signal processing. They are particularly useful when dealing with high-dimensional data or when the underlying structure of the data needs to be uncovered.