**Word Representation:**

**1. Distributional Semantics**

* Understanding word meaning based on context.
* Three approaches to semantics:
  1. **Ontological Semantics**
  2. **Decompositional Semantics**
  3. **Distributional Semantics** (focus of the discussion)
* J.R. Firth (1957) quote: "You shall know a word by the company it keeps."

**2. Count-Based Approaches to Word Representation**

* Traditional NLP methods relied on frequency-based models.
* Word meaning inferred from neighboring words.
* Example: Different meanings of "banking" based on context.

**3. Term-Context Matrix**

* Constructing a **term-context matrix** (T-context matrix):
  + Rows: Individual words (terms)
  + Columns: Context words
  + Entry values: Number of times a term appears with a context word.
* Difference from bigram models:
  + Bigram models consider only adjacent words.
  + Term-context models consider a fixed window size (e.g., 10 words).
* Words with similar context vectors likely have similar meanings.

**4. Challenges of Term-Context Matrix**

* **High dimensionality:** Large vocabulary results in large vectors.
* **Sparsity:** Most matrix entries are zero.
* **Bias towards frequent words:** Stop words have high counts.
* **Need for normalization** to improve effectiveness.

**5. TF-IDF (Term Frequency - Inverse Document Frequency)**

* **TF (Term Frequency):**
  + Raw count of word occurrences in a document.
  + Log transformation applied: log(count + 1) to smooth values.
* **Problems with TF:**
  + Common words (stop words) still have high values.
* **IDF (Inverse Document Frequency):**
  + Measures word importance by considering its rarity across documents.
  + Formula: log(N / DF), where:
    - N = total number of documents
    - DF = number of documents containing the word
  + Penalizes common words while highlighting rare but significant words.
* **TF-IDF Calculation:** TF × IDF
* **Advantages of TF-IDF:**
  + Reduces impact of frequently occurring but non-informative words.
  + Emphasizes words that are important to specific documents.
* **Example Comparison:**
  + "Romeo" has high TF-IDF (appears in few documents, highly specific).
  + "Good" has low TF-IDF (common across documents, less significant).

**6. Applications of Distributional Semantics and TF-IDF**

* Used in **document retrieval** and **search engines**.
* Helps in **document classification** and **word similarity analysis**.
* Forms basis for more advanced techniques like **Word Embeddings** (Word2Vec, GloVe, etc.).

**Summary**

* **Distributional Semantics** helps in understanding word meaning via context.
* **Term-context matrix** provides a way to quantify word relationships.
* **Challenges** include high dimensionality and sparsity.
* **TF-IDF** improves traditional models by emphasizing important words.
* **TF-IDF is widely used in information retrieval and text analysis.**

This covers all key concepts from the transcript in a concise manner for quick revision.

**Prediction-Based Word Embedding Models: CBOW & Skip-Gram**

**1. Introduction to Word Embeddings**

* **Word embeddings capture the semantic meaning of words by mapping them to vectors.**
* **Continuous Bag of Words (CBOW) and Skip-Gram are two popular prediction-based models for word embedding.**

**2. CBOW vs. Skip-Gram**

* **CBOW (Continuous Bag of Words): Uses surrounding context words to predict the target word.**
* **Skip-Gram: Uses the central word to predict surrounding context words.**
* **Both models use a neural network framework to learn word representations.**

**3. Skip-Gram with Negative Sampling (SGNS)**

* **Skip-Gram is more commonly used as it performs better with small datasets and infrequent words.**
* **Negative sampling is used to optimize computation by distinguishing positive word pairs from negative samples.**

**4. Unsupervised and Self-Supervised Learning**

* **Word2Vec is an unsupervised learning approach, specifically a self-supervised method, as it doesn’t require labeled data.**
* **It is framed as a binary classification task (logistic regression) to determine if a given word pair belongs to the same context window.**

**5. Concept of Target and Context Words**

* **Target Word (T): The central word in the sliding window.**
* **Context Words (C): The surrounding words within a fixed window size.**
* **Positive context words are those appearing within the window, while non-context (negative) words are sampled from the corpus.**

**6. Context Window and Word Pairs**

* **A fixed-size window (e.g., size 10) determines the number of context words around the target.**
* **For every target word, positive pairs are constructed with its neighboring words.**
* **Negative samples (non-context words) are randomly selected from the corpus, typically in a higher ratio (e.g., 4:1 or 5:1).**

**7. Training Process: Skip-Gram Model**

* **The model learns embeddings through a classification task:**
  + **Given a word pair (T, C), predict if C is a true context word.**
* **Uses a logistic regression classifier with a sigmoid activation function to measure the probability of a word pair being positive.**
* **The classifier is trained using a log-likelihood loss function.**

**8. Probability Computation**

* **The probability of a word pair (T, C) being positive is computed using:**

**P(C∣T)=σ(C⋅T)=11+e−C⋅TP(C|T) = \sigma(C \cdot T) = \frac{1}{1 + e^{-C \cdot T}}**

* **Negative probability: 1−P(C∣T)1 - P(C|T)**
* **The loss function maximizes the log-likelihood of positive pairs while minimizing it for negative pairs.**

**9. Negative Sampling**

* **Instead of considering all words in the vocabulary, a subset of negative samples is used.**
* **Negative sampling helps in reducing computation and improving learning efficiency.**
* **Typically, 4 to 5 times more negative samples than positive samples are used.**

**10. Optimization and Training**

* **Objective: Maximize log-likelihood estimation for positive pairs and minimize for negative pairs.**
* **Gradient descent is used to update word vectors iteratively.**
* **Each word has two embeddings: one when it acts as a target and another when it acts as a context word.**

**11. Final Word Representations**

* **After training, each word is associated with a dense, continuous vector.**
* **These embeddings capture semantic similarities, e.g., ‘king - man + woman ≈ queen’.**
* **Pre-trained embeddings can be used in various NLP applications such as text classification, machine translation, and sentiment analysis.**

**12. Summary**

* **CBOW and Skip-Gram are fundamental models for word embeddings.**
* **Skip-Gram with negative sampling optimizes training efficiency.**
* **The embeddings learned represent words in a meaningful way, enabling NLP tasks to capture contextual and semantic information effectively.**

**Training Process**

1. **Defining the Training Objective**
   * **Each word has a target word and a context window around it.**
   * **Positive samples: Valid (word, context) pairs from the dataset.**
   * **Negative samples: Randomly sampled words that do not appear in the context window.**
2. **Negative Sampling Process**
   * **Negative samples are drawn from a unigram distribution.**
   * **A smoothing factor α (alpha) is applied to boost rare words' probabilities.**
   * **Example: If A appears 99% and B 1%, modifying with α = 0.75 results in A = 97% and B = 3%, giving rare words a better chance to be sampled.**
3. **Training with Logistic Regression**
   * **The goal is to:**
     + **Maximize the probability of positive samples (target word & valid context).**
     + **Maximize the probability that negative samples are actually negative.**
   * **Uses sigmoid activation to compute probabilities.**
4. **Loss Function**
   * **Objective function:**

**log⁡P(w,cpos)+∑i=1Klog⁡P(w,cneg)\log P(w, c\_{pos}) + \sum\_{i=1}^{K} \log P(w, c\_{neg})logP(w,cpos​)+i=1∑K​logP(w,cneg​)**

* + **Uses gradient descent to update word vectors.**

1. **Final Word Vectors**
   * **Each word has two embeddings (when acting as a target or context word).**
   * **The final word vector is often the sum of both.**

**Comparison: Count-Based vs Prediction-Based Approaches**

| **Feature** | **Count-Based (TF-IDF, Co-occurrence Matrix)** | **Prediction-Based (Word2Vec)** |
| --- | --- | --- |
| **Data Requirement** | **One-time pass through data** | **Iterative learning** |
| **Model Type** | **Stores word frequencies** | **Learns embeddings** |
| **Memory Usage** | **Large matrix** | **Smaller vectors** |
| **Adaptability** | **Requires recomputation** | **Can learn continuously** |

**Subsampling High-Frequency Words**

* **Words like "the," "is," "a" appear frequently but provide less meaning.**
* **The paper introduces subsampling:**
  + **Words with high frequency are removed with a certain probability.**
  + **The probability is determined by:**

**P(w)=1−tf(w)P(w) = 1 - \sqrt{\frac{t}{f(w)}}P(w)=1−f(w)t​​**

**where t is a threshold (e.g., 10⁻⁵), and f(w) is the word frequency.**

* + **Effect: Rare words are sampled more, frequent words less.**

**Key Takeaways**

* **Word2Vec is a prediction-based embedding model that learns word vectors through context prediction.**
* **Negative sampling helps avoid computing full softmax probabilities.**
* **Subsampling high-frequency words improves training efficiency.**
* **Final word embeddings are static, unlike modern contextual embeddings (e.g., BERT).**