Università degli studi di Trento

Natural Language Understanding

Prof. Riccardi

Academic Year 2024/2025

Contents

Chapter 1

Basic Concepts

1.1 The Building Blocks of Speech

Human language, when spoken, exists as a continuous acoustic signal—a waveform. When we analyze this waveform using techniques like computing a **spectrogram**, we visualize the distribution of energy across different frequencies over time. This analysis reveals patterns and structures within the speech signal that correspond to discrete units of sound.

These fundamental units of sound are called **phonemes**. Phonemes are the smallest units of sound in a language that can distinguish one word from another. For example, the sounds /p/ and /b/ in English are phonemes because they differentiate words like 'pat' and 'bat'. While we use letters in writing, in phonology, we use symbols from the **International Phonetic Alphabet (IPA)** to represent these speech sounds precisely, ensuring a consistent and universal way to transcribe the sounds of any language.

Phonemes can be further described in terms of their **distinctive features**. These are articulatory or acoustic properties that group phonemes into natural classes and help explain phonological patterns. Distinctive features essentially parametrize the physical process of generating speech in the vocal tract. They also create a 'similarity space' for speech sounds, indicating which sounds are more alike based on shared features.

Examples of distinctive features include:

- **Voicing**: This feature distinguishes sounds produced with vibration of the vocal cords from those produced without vibration. Spectrograms often show different patterns for voiced and unvoiced sounds.
- **Place of Articulation**: This describes where in the vocal tract a sound is produced.
- **Manner of Articulation**: This describes how the airflow is obstructed or modified to produce a sound.

Understanding these features is crucial because they influence how sounds behave in different contexts within a language.

1.2 Levels of Linguistic Structure

Beyond individual phonemes, language is organized into hierarchical structures. We can identify several key levels:

- **Segmental Structure**: This refers to the linear sequence of phonemes (segments) that make up words and utterances. It deals with the individual consonant and vowel sounds.
- **Morphophonology**: This level examines the interaction between morphology (the study of word structure) and phonology (the study of sound systems). It looks at how the pronunciation of morphemes (meaningful units, like roots and affixes) can change when they combine.
- **Syllabic Structure**: Phonemes are organized into syllables, which are fundamental units of pronunciation. A typical syllable can be broken down into:
  - *Nucleus*: The central, most sonorous part of the syllable, usually a vowel.
  - *Onset*: The consonant(s) preceding the nucleus (optional in some languages).
  - *Coda*: The consonant(s) following the nucleus (optional in many languages).
- **Prosodic Structure**: This level goes beyond the individual segments and syllables to encompass features like stress, intonation, and rhythm. It's not just *what* you say, but *how* you say it.
  - *Stress*: A relative prominence placed on a syllable or word, often realized through higher pitch, greater intensity, or longer vowel duration. Stress can differentiate words and highlight important information in a sentence.
  - *Intonation*: The rise and fall of pitch across an utterance, which can convey grammatical information (e.g., distinguishing a statement from a question) or emotional attitude.
  - *Rhythm*: The pattern of stressed and unstressed syllables.

Prosodic structure is vital for conveying meaning and intent. A speaker can change the prosody to alter the intended meaning of a sentence, even if the words themselves remain the same.

Understanding these levels of linguistic structure, including their symbolic representation and the constraints on how units combine at each level, is essential for Natural Language Understanding. It allows for better error analysis in NLP systems, informs the interpretation and evaluation of their outputs, and provides valuable knowledge that can be incorporated into machine learning algorithms.

1.3 The Structure of Language: Competence and Performance

In linguistics, particularly following the work of Noam Chomsky, a crucial distinction is made between **linguistic competence** and **linguistic performance**.

Chomsky (1965) described linguistic theory as being primarily concerned with:

> "... an ideal speaker-listener, in a completely homogeneous speech-community, who knows its language perfectly and is unaffected by such

grammatically irrelevant conditions as memory limitations, distractions, shifts of attention and interest, and errors (random or characteristic) in applying his knowledge of the language in actual performance."

- **Linguistic Competence**: This refers to an individual's underlying, internalized knowledge of their language's grammar and vocabulary.
- **Linguistic Performance**: This is the actual use of language in concrete situations.

In essence, competence is the *knowledge* of the language, while performance is the *actual use* of the language.

Computational Linguistics can be seen as occupying a position somewhat in between competence and performance. While theoretical linguistics often idealizes language to study competence, computational linguistics must grapple with the messy reality of performance – the language as it is actually used, with all its disfluencies and variations. However, computational linguistics also aims to build models that reflect and utilize aspects of linguistic competence to process and understand language effectively. It seeks a **"third way"** to connect the theoretical understanding of competence with the practical analysis of performance, potentially leading to theories of competence that can account for phenomena observed in actual language use (like hesitations and repetitions).

1.4 Natural Language in Communication and Interaction

Language is fundamentally a tool for communication and interaction between individuals. This involves distinct roles:

- **Speaker (or Writer)**: The individual who produces the message. The speaker makes an effort, engaging mental and motor resources, to formulate and communicate a message. Speakers often strive to communicate effectively while minimizing their own effort.
- **Hearer (or Reader)**: The individual who receives and interprets the message. The hearer makes an effort, engaging mental and perceptual resources, to decode and understand the message, ideally resolving any ambiguity.

Successful communication relies on both participants collaborating to ensure the message is conveyed and understood in context.

1.4.1 Language Diversity

The world's languages exhibit remarkable diversity. Linguists classify languages in several ways:

- **Genealogical Classification**: Groups languages based on shared ancestry.
- **Aerial Classification**: Groups languages based on geographical proximity and language contact.
- **Typological Classification**: Groups languages based on shared structural characteristics, regardless of historical relationship or geographical location.

One well-known typological parameter is **Word Order**. While languages exhibit a variety of word orders, Subject-Verb-Object (SVO) and Subject-Object-Verb (SOV) orders are the most common globally, accounting for a large majority of languages.

1.4.2 Head Directionality

Another important typological parameter is **Head Directionality**. This refers to the order of the "head" of a phrase relative to its dependents. The head is the core element that determines the category of the phrase (e.g., the noun in a noun phrase, the verb in a verb phrase).

- **Head-Initial (Right-Branching)**: The head precedes its dependents. English is largely head-initial (e.g., Verb precedes Object: "eat **an apple**", Preposition precedes Noun Phrase: "**in** the garden").
- **Head-Final (Left-Branching)**: The head follows its dependents. Japanese is a consistently head-final language (e.g., Object precedes Verb: "**ringo o taberu**" (apple object eat)).

Understanding head directionality helps explain structural differences across languages and is important for parsing and generating text in different languages.

1.5 Towards Dialogue and Discourse

Analyzing language solely at the sentence level is often insufficient for true understanding. Natural language exists within a broader context of **discourse**, which refers to sequences of language units (sentences, turns in a conversation) that are connected and meaningful. Understanding discourse requires considering how sentences relate to each other and the shared context between communicators. Key aspects of discourse include:

- **Information Structure**: This concerns how the speaker organizes the information within a sentence based on the presumed knowledge of the hearer and the flow of the conversation. It involves distinguishing between information that is already known or inferable (the **topic** or **given** information) and information that is new or being highlighted (the **focus** or **new** information). This influences sentence structure and prosody.
- **Focus and Presupposition**:
  - **Focus**: The part of a sentence that conveys the most important or new information the speaker intends to communicate. Highlighting the focus can change the meaning or implication of a sentence.
  - **Presupposition**: Information that the speaker assumes is already known or accepted by the hearer. Presuppositions are taken for granted in the utterance. Understanding what is presupposed is crucial for correct interpretation.
- **Adjacent Sentence Relationships**: Consecutive sentences in a text or dialogue are rarely isolated. They are connected by **discourse relations** (also called coherence relations). These relations specify how the meaning of one

sentence relates to the meaning of the preceding one. Examples include: Temporal sequence (event A followed by event B), Cause-and-effect (event A caused event B), Elaboration (sentence B provides more detail about sentence A), Contrast (sentence B presents information contrary to sentence A). Often, these connections are not explicitly marked by conjunctions and must be inferred by the listener or reader.

- **Beyond Adjacent Sentences**: Discourse coherence extends beyond immediate sentence pairs. Relationships can exist between sentences that are further apart, forming larger segments of the discourse. In conversations, these can be local within a sub-dialogue or global across the entire interaction. In written documents, paragraphs and sections are linked in meaningful ways.
- **Natural Language Documents**: While dialogue provides a rich context for language study, analyzing longer written documents is also critical for NLU. Financial reports, legal contracts, medical records, essays, and social media feeds are all forms of natural language data with their own structures, conventions, and challenges for automatic processing and understanding.

Analyzing language at the discourse level is essential for tasks like summarization, question answering, machine translation, and building conversational agents, as the meaning of individual sentences is often dependent on the surrounding linguistic context.

Chapter 2

Language Modeling

Language modeling is a core aspect of Natural Language Processing (NLP) that focuses on predicting the **probability distribution** of sequences of words. In simpler terms, a language model estimates the likelihood of a sequence of words occurring or predicts the probability of the next word given the preceding words.

The problem of language modeling arises because natural language, unlike random sequences, exhibits structure and predictable patterns. The goal is to learn these patterns from data to build models that can assign probabilities to word sequences.

2.1 Statistical Language Models

Statistical Language Models (SLMs) approach language modeling by estimating the probability of a sequence of words based on the frequency of word occurrences in large text corpora. These models rely on probability distributions derived directly from observed data.

2.1.1 The Language Modeling Problem and Chain Rule

The fundamental problem is to compute the probability of an entire sequence of words $W = (w_1, w_2, \ldots, w_N)$, denoted as $P(W)$. Using the Chain Rule of Probability, this joint probability can be decomposed into a product of conditional probabilities:

$$P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_N|w_1, \ldots, w_{N-1})$$

This can be written more compactly as:

$$P(W) = \prod_{i=1}^{N} P(w_i|w_1, \ldots, w_{i-1})$$

where $P(w_i|w_1, \ldots, w_{i-1})$ is the probability of word w_i given the entire preceding sequence of words $w_1, \ldots, w_{i-1}$.

2.1.2 N-gram Models

Calculating the probability of a word given the entire history of preceding words is computationally expensive and requires an enormous amount of data to estimate reliably. To make this tractable, N-gram models introduce the **Markov Assumption**. The assumption is that the probability of the next word depends only on the preceding $N - 1$ words, rather than the entire history.

$$P(w_i|w_1, \ldots, w_{i-1}) \approx P(w_i|w_{i-(N-1)}, \ldots, w_{i-1})$$

N-gram models, particularly bigram and trigram models, were historically the most widely used statistical language models due to their simplicity and relative effectiveness in capturing local word dependencies. They estimate the conditional probabilities $P(w_i|w_{i-(N-1)}, \ldots, w_{i-1})$ directly from the frequencies of word sequences in a training corpus.

2.1.3 Maximum Likelihood Estimation (MLE)

The standard way to estimate the probabilities in N-gram models from a corpus is using Maximum Likelihood Estimation (MLE). For an N-gram $w_{i-(N-1)}, \ldots, w_i$, the MLE probability is calculated as the frequency of the N-gram divided by the frequency of the $(N - 1)$-gram prefix:

$$P_{MLE}(w_i|w_{i-(N-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(N-1)}, \ldots, w_i)}{\text{count}(w_{i-(N-1)}, \ldots, w_{i-1})}$$

These counts are obtained from the training corpus. This estimation can be visualized using a co-occurrence matrix, where entries represent the frequency of word pairs.

2.1.4 Data Sparsity and Smoothing Techniques

A major limitation of N-gram models, especially for higher values of N, is **data sparsity**. In any finite corpus, many perfectly valid word sequences (N-grams) will not appear even once. Using MLE, any N-gram that does not appear in the training data will be assigned a probability of zero. This is problematic because a single zero probability in the product calculated by the Chain Rule results in the entire sequence having a probability of zero, which is unrealistic for unseen but possible sequences. This sparsity also makes it difficult for N-grams to capture long-range dependencies effectively, as the context window is limited to $N - 1$ words.

To address data sparsity and prevent zero probabilities, **smoothing techniques** are employed. Smoothing adjusts the MLE probabilities, typically by slightly decreasing the probability of seen N-grams and redistributing that probability mass to unseen N-grams. Common methods include:

- **Laplace Smoothing (Add-one Smoothing)**: This is the simplest smoothing method. It adds a small constant (typically 1, hence "add-one") to all N-gram counts before normalizing. This ensures that no N-gram has a count of zero. While easy to compute, add-one smoothing often adds too much probability mass to unseen events, significantly distorting the probabilities of frequently seen events. More generally, Add-k smoothing adds a constant k where $0 < k < 1$.

- **Good-Turing Smoothing**: Based on the intuition of estimating the probability of encountering an unseen event (like catching a new species of fish), Good-Turing re-estimates the counts of items with frequency r based on the number of items that occurred $r + 1$ times. This provides a more sophisticated way to allocate probability mass to unseen events ($r = 0$) and adjust the counts of observed events.
- **Backoff and Interpolation**: These strategies combine probability estimates from different N-gram orders.
  - **Backoff**: If there is enough data to estimate a higher-order N-gram (e.g., trigram), use its probability. If not (count is zero or below a threshold), "back off" to a lower-order N-gram (e.g., bigram), possibly applying a discount. If the bigram count is also insufficient, back off to a unigram. A normalizing factor (alpha) is used to ensure probabilities sum to one. Katz's backoff is a well-known algorithm.
  - **Interpolation**: This method always combines estimates from multiple N-gram orders by taking a weighted average. The weights are typically learned from held-out data.
- **Kneser-Ney Smoothing**: It adjusts probabilities by considering not just how often an N-gram occurs, but also how likely its components are to appear in novel contexts.

Handling **Out-Of-Vocabulary (OOV) words** (words not seen in the training data) is related to smoothing. A common approach is to replace rare words in the training data with a special unknown word token, <UNK>, and then train the language model including this token. This allows the model to assign a non-zero probability to unseen words encountered during testing or deployment.

2.1.5 Cache Language Models

An interesting idea to leverage recent history beyond the fixed N-gram window is the concept of **Cache Language Models**. The intuition is that words that have appeared recently in the discourse are more likely to reappear soon. A cache LM maintains a memory (a buffer) of recent words and incorporates a probability estimate based on this cache, interpolating it with the standard N-gram probability.

2.1.6 Semantic Language Models

SLMs attempt to incorporate semantic information or the meaning of words and phrases into the prediction process. This allows the model to potentially capture longer-range dependencies related to topics or concepts in the discourse, even if the words are not syntactically related. Deep neural networks have been explored for building semantic language models.

2.2 Neural Language Models

Neural Language Models (NLMs) utilize neural networks, particularly deep learning techniques, to address some of the limitations of statistical models. They learn to represent words and contexts in continuous vector spaces (embeddings), allowing them to capture more complex patterns and potentially better handle long-range dependencies and data sparsity implicitly compared to traditional N-grams with smoothing.

2.2.1 Feedforward Neural Networks

Early neural language models used feedforward neural networks (FNNs). While they can learn effective word representations and context, they still operate on a fixed-size context window, similar to N-grams, which limits their ability to capture long-range dependencies.

2.2.2 Recurrent Neural Networks (RNNs)

Unlike feedforward networks, RNNs have recurrent connections that allow information to persist across time steps. They maintain a hidden state (or context layer) that acts as a memory, theoretically allowing them to process and be influenced by the entire preceding sequence, not just a fixed window. This enables them to capture longer-range dependencies.

However, standard RNNs can suffer from issues like the vanishing gradient problem, which makes it difficult to learn long-term dependencies effectively in practice. Variants like **Long Short-Term Memory (LSTM)** networks and **Gated Recurrent Units (GRUs)** were developed to mitigate this problem through gating mechanisms that control the flow of information, significantly improving their performance on sequential tasks.

2.2.3 Transformer Models and BERT

The introduction of the **Transformer** architecture, notably with models like **BERT** (Bidirectional Encoder Representations from Transformers), revolutionized NLP and language modeling. Transformers rely heavily on **self-attention mechanisms**. Self-attention allows the model to weigh the importance of different words in the input sequence when processing each word, effectively capturing dependencies regardless of their distance in the sequence (global dependencies).

2.2.4 Large Language Models (LLMs)

Building on the success of the Transformer architecture and trained on massive datasets, **Large Language Models (LLMs)** have emerged as a dominant force in modern NLP. Models like GPT-3 exemplify this trend, trained on hundreds of billions of tokens from diverse internet text and books. The scale of these models (billions or trillions of parameters) and their training data allows them to exhibit remarkable capabilities in text generation, question answering, and understanding, often performing well even on tasks they were not explicitly fine-tuned for. However, LLMs also present significant challenges, including high computational costs, potential for inheriting and amplifying biases from their vast training data, and the phenomenon of "hallucination," where they generate plausible-sounding but factually incorrect or nonsensical information.

2.3 Evaluation Metrics

Evaluating the quality and effectiveness of language models is crucial. Several metrics are used, depending on the model type and the intended application.

2.3.1 Perplexity

Perplexity (PP) is a widely used intrinsic metric for evaluating language models, especially predictive models like N-grams and RNNs. It measures how well a probability model predicts a sample of text. A lower perplexity score indicates that the model is better at predicting the test data, meaning the test data is less "perplexing" to the model.

Perplexity is closely related to **cross-entropy**. For a test set $W = (w_1, w_2, \ldots, w_N)$, the cross-entropy $H(W)$ of a language model q (our model) with respect to the true distribution p (of the language, which we don't know) can be approximated based on the model's probability assignments to the words in the test set:

$$H(W) \approx -\frac{1}{N}\sum_{i=1}^{N} \log_2 q(w_i|w_1, \ldots, w_{i-1})$$

Perplexity is then defined as 2 raised to the power of the cross-entropy:

$$\text{PPL}(W) = 2^{H(W)} = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 q(w_i|w_1,\ldots,w_{i-1})}$$

Perplexity can also be interpreted as the weighted average number of choices the model has for the next word. For example, a perplexity of 10 for a vocabulary of 30,000 suggests that, on average, the model is as "uncertain" about the next word as if it were choosing from 10 options with equal probability, even though the actual vocabulary size is much larger.

2.3.2 BLEU and ROUGE Scores

For evaluating the output of generative language models in specific tasks, extrinsic metrics are often used:

- **BLEU (Bilingual Evaluation Understudy)**: This metric is primarily used for evaluating machine translation. It measures the similarity between the generated translation and one or more human-created reference translations, based on the precision of N-grams.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**: This metric is commonly used for evaluating text summarization. It measures the overlap (based on recall) of N-grams, word sequences, or pairs of words between the generated summary and human reference summaries.

2.4 Applications and Challenges

Language models are fundamental components in a wide array of NLP applications but also face significant challenges.

2.4.1 Applications

The ability to model word sequences and predict upcoming words makes language models indispensable in applications such as:

- **Automatic Speech Recognition (ASR)**: LMs help select the most likely sequence of words corresponding to an acoustic signal by providing probabilities for different word combinations.
- **Machine Translation**: LMs are used to evaluate the fluency and grammatical correctness of potential translations.
- **Automatic Text Generation**: Creating human-like text for chatbots, content creation, and story generation relies heavily on LMs to produce coherent and probable word sequences.
- **Autocomplete and Spelling Correction**: Predicting the next word a user intends to type or suggesting correct spellings for misspelled words leverages the probabilistic nature of LMs.
- **Information Retrieval and Extraction**: LMs can help understand queries and documents.
- **Sentiment Analysis**: Probabilities of certain word sequences can correlate with sentiment.

2.4.2 Challenges

Despite their success, language models face several important challenges:

- **Data Bias**: Language models learn from the data they are trained on. If this data contains societal biases (e.g., related to gender, race, or occupation), the model will likely learn and perpetuate these biases in its predictions and generations, potentially leading to unfair or unethical outcomes.
- **Computational Cost**: Training large-scale neural language models, especially LLMs, requires immense computational resources and energy consumption, making them expensive to develop and deploy.
- **Hallucination and Misinformation**: Advanced neural models can sometimes generate fluent and convincing text that is factually incorrect or nonsensical – a phenomenon known as hallucination. This poses a significant challenge, particularly when models are used to generate factual information, and recent research explores why this occurs.
- **Interpretability**: Understanding *why* a complex neural language model makes a specific prediction is often difficult, making them less transparent than statistical models.
- **Handling Rare and Out-Of-Vocabulary Words**: While smoothing and UNK tokens help, effectively handling words and phrases that are very rare or completely absent from training data remains a challenge, particularly in domains with specialized vocabulary.

- **Evaluation Limitations**: Intrinsic metrics like perplexity have limitations and may not fully reflect performance on complex downstream tasks. Extrinsic evaluations on specific tasks are necessary but can be time-consuming.

Chapter 3

Large Language Models

Large Language Models (LLMs), also often referred to as **Foundation Models**, are advanced artificial intelligence models trained on vast amounts of broad data. These models are typically trained using self-supervision at scale and can be adapted to a wide range of downstream tasks. LLMs are designed to understand, generate, and process human language, enabling them to perform activities such as generating text, translating languages, writing different types of creative content, and answering questions in an informative way.

3.1 How They Work

LLMs leverage deep learning techniques, primarily based on the **Transformer architecture**. The training process involves pre-training on enormous and diverse datasets. Through this training, LLMs learn complex patterns, statistical relationships, and underlying structures within the text data, enabling them to predict the next word or sequence of words given a preceding context.

LLMs have demonstrated remarkable **capabilities** across a wide range of natural language tasks. Some key abilities include: Text Generation, Language Translation, Question Answering, Text Summarization, Code Generation, Completing Sentences or Sequences. These capabilities are constantly evolving with the development of newer and larger models.

3.2 Evaluation of Large Language Models

Evaluating the performance of LLMs is crucial but also challenging. Traditional language model evaluation metrics like perplexity, which measure how well a model predicts the next word, are often insufficient for LLMs in their current form. This is because perplexity does not directly assess the factual correctness or the semantic quality of the generated text.

Therefore, the evaluation of LLMs increasingly relies on task-specific and benchmark-based assessments that aim to ground the evaluation in the ”truth” or semantics of the task.

3.2.1 Benchmarks and Evaluation Methods

Various benchmarks have been developed to evaluate different aspects of LLMs' capabilities:

- **MMLU (Massive Multitask Language Understanding)**: Evaluates a model's knowledge across a wide range of subjects (57 different areas), including STEM, humanities, social sciences, etc., typically using multiple-choice questions.
- **ARC Challenge (AI2 Reasoning Challenge)**: Focuses on science questions requiring reasoning, often aimed at grade-school level understanding.

However, even models that perform well on benchmarks can exhibit failures. For instance, models might struggle with simple negations in questions or fail to correctly perform multi-step reasoning or equation solving, sometimes producing elaborate but incorrect step-by-step solutions or even indicating inability to solve the problem when it is solvable.

Beyond standardized benchmarks, other evaluation methods include:

- **User-based evaluations**: Platforms like Chatbot Arena allow users to compare the outputs of different models side-by-side without knowing which model produced which response, providing subjective feedback. While indicative of user preference, this method is less scientifically controlled.
- **Task-specific evaluations**: Evaluating the model's performance directly on downstream tasks like summarization, translation, or dialogue requires task-specific metrics (like ROUGE, BLEU, or human judgment).

3.3 Challenges and Limitations

Despite their impressive capabilities, LLMs still face significant challenges and limitations:

- **Data Bias**: LLMs learn from the data they are trained on, which often reflects societal biases. Consequently, models can inherit and amplify these biases, leading to prejudiced or unfair outputs.
- **Hallucination and Misinformation**: LLMs can generate outputs that are fluent and convincing but factually incorrect, nonsensical, or detached from reality. This "hallucination" is a major concern, especially when models are used to generate factual information, and it relates to issues of generating false or misleading content.
- **Lack of True Commonsense or World Knowledge**: While they exhibit impressive abilities, LLMs do not possess genuine understanding, consciousness, or commonsense reasoning in the human sense. Their knowledge is statistical and derived from correlations in the training data, which can lead to errors in situations requiring deep understanding or reasoning about the physical world.
- **High Computational Costs**: Training and running LLMs, especially the largest ones, requires substantial computational resources, energy, and infrastructure, making them expensive and less accessible.
- **Scalability and Efficiency**: While scaling laws suggest performance im-

provements with increased model size and data, there are practical limits and efficiency challenges.

- **Outstanding Issues**: Beyond performance, there are significant concerns regarding:
  - **Harm (Health and Societal)**: The potential for LLMs to be used to generate harmful content, spread disinformation, or have negative societal impacts.
  - **Privacy**: Risks associated with handling sensitive user data and the potential for models to inadvertently leak information from their training data.
  - **Cybersecurity**: Potential vulnerabilities of LLMs to adversarial attacks and their possible misuse in cybercrimes.

Understanding these challenges is critical for the responsible development and deployment of Large Language Models.

Chapter 4

Distributional Semantics

4.1 Distributional Semantics and the Theoretical Hypothesis

Distributional semantics is an approach in computational linguistics and natural language processing that characterizes the meaning of words based on their usage in large text corpora. At its core is the **Theoretical Hypothesis**, which states: *"Linguistic items with similar distributions have related meanings (behave similarly) in similar contexts."* In essence, a word's meaning is determined by the company it keeps – the words and contexts in which it appears.

This hypothesis is supported by the observation that words can take on different senses depending on their context. Similarly, words with related meanings, like "science" and "physics", tend to appear in similar contexts, although they are not perfect synonyms and their behavior may diverge in specific cases.

4.2 Word Representations

To leverage the distributional hypothesis computationally, we need to represent words in a format that allows us to measure and compare their distributions. **Vector semantics** achieves this by representing words as mathematical vectors.

4.2.1 Sparse Representations

- **One-hot encoding**: In this simple method, each word in the vocabulary is assigned a unique index and represented as a high-dimensional vector (with dimensionality equal to the vocabulary size). All dimensions are zero except for the dimension corresponding to the word's index, which is set to one. This results in very sparse vectors. While easy to create, one-hot encoding captures no inherent linguistic knowledge or similarity between words.
- **Frequency-Based Vector Space Models (VSMs)**: These models represent words based on their co-occurrence statistics within a corpus.
  - **Term-Document Matrices**: Words (terms) are represented as vectors based on their frequency of occurrence in different documents. The ma-

trix has words as rows and documents as columns.

  - **Word-Word Co-occurrence Matrices**: Words are represented based on how often they co-occur with other words within a defined context window. The matrix has words as both rows and columns.

These frequency-based matrices are typically high-dimensional and sparse, especially for large vocabularies and corpora.

4.2.2 Weighting Schemes for Frequency-Based Models

Raw frequency counts can be problematic as very frequent words (like "the" or "a") may dominate the vectors and not be very discriminative of a word's specific meaning. Weighting schemes are used to give more importance to words that are more informative:

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: A common weighting scheme for term-document matrices. It is the product of two terms:
  - *Term Frequency (TF)*: Measures how often a term appears in a document. Often a log-scaled count is used: $\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t, d) + 1)$.
  - *Inverse Document Frequency (IDF)*: Measures how rare a term is across all documents. Terms that appear in fewer documents are given higher weight: $\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$, where N is the total number of documents and df_t is the number of documents containing term t.

  The TF-IDF weight is $\mathrm{wt}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$. This weighting balances the frequency of a term within a document with its rarity across the corpus.
- **PPMI (Positive Pointwise Mutual Information)**: A weighting scheme often used for word-word co-occurrence matrices to measure the association between two words, normalizing for their individual frequencies. Positive PMI is used to avoid negative values.

4.2.3 Word Clusters

Instead of dense vectors, words can also be grouped into clusters based on their distributional properties. **Brown Clusters** are an example of a hierarchical word clustering algorithm that groups words based on the distribution of words preceding and following them. This results in a tree-like structure where each node represents a cluster, and words are leaves, often represented by binary strings corresponding to paths in the tree.

4.2.4 Dense Representations (Word Embeddings)

Unlike sparse, high-dimensional representations, **distributed representations** or **word embeddings** are dense, lower-dimensional vectors with continuous values. These representations aim to capture semantic and syntactic relationships such that words with similar meanings are located close to each other in the vector space. The values in these vectors are typically learned from large text corpora using neural networks or matrix factorization techniques. Dense representations are often more computationally efficient and can capture more nuanced relationships than sparse count-based vectors.

4.3 Word Similarity and Distance Metrics

To measure the similarity between word vectors in a vector space, we use distance or similarity metrics:

- **Cosine Similarity**: The most common metric. It measures the cosine of the angle between two vectors. The cosine similarity between vectors $\mathbf{v}$ and $\mathbf{w}$ is calculated using their dot product and magnitudes:

$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{||\mathbf{v}|| \, ||\mathbf{w}||} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

  Cosine similarity is preferred over the raw dot product because it normalizes for the length of the vectors, preventing high-frequency words (which tend to have longer vectors) from artificially appearing more similar. For vectors that have been normalized to unit length, the dot product is equivalent to the cosine similarity.
- **Euclidean Distance**: Measures the straight-line distance between two points (vectors) in the space: $||\mathbf{v} - \mathbf{w}|| = \sqrt{\sum_{i=1}^{N} (v_i - w_i)^2}$. Smaller distance indicates higher similarity.

4.4 Static Word Embeddings

Static word embeddings learn one fixed vector representation for each word in the vocabulary, regardless of its context.

4.4.1 Word2Vec

Word2Vec is a highly influential method for learning static word embeddings using shallow neural networks. The core idea is to train a model to "predict" neighboring words, rather than just counting co-occurrences. It uses a self-supervised approach, leveraging the running text itself as supervision signal. Word2Vec has two main architectures:

- **Continuous Bag of Words (CBOW)**: Predicts a target word based on the average of the vectors of the words in its surrounding context window.
- **Skip-gram**: Predicts the context words within a window given the target word. This architecture, especially with negative sampling, is often more effective for learning representations of rare words.

Word2Vec is known for its speed and its ability to capture surprising semantic and syntactic regularities.

4.4.2 GloVe (Global Vectors for Word Representation)

GloVe learns representations by factorizing a global word-word co-occurrence matrix. It aims to capture the ratio of co-occurrence probabilities, which are argued to be more informative about word relationships than probabilities themselves.

4.4.3 FastText

FastText is an extension of Word2Vec developed by Facebook. Its key innovation is that it represents each word as a bag of character n-grams. The vector for a word is the sum of the vectors for its character n-grams. This allows FastText to:

- Handle out-of-vocabulary (OOV) words by composing vectors for their n-grams.
- Learn better representations for rare words, as their n-grams may appear in other words.
- Capture morphological similarities between words.

4.5 Context-Based (Contextualized) Embeddings

Unlike static embeddings, **context-based embeddings** produce word vectors that are a function of the entire input sentence.

- **ELMo (Embeddings from Language Models)**: ELMo generates contextualized embeddings using a deep, bidirectional LSTM network trained on a language modeling objective. The final embedding for a word is a linear combination of the representations from all layers of the LSTM, allowing downstream tasks to learn which layers are most useful.
- **BERT (Bidirectional Encoder Representations from Transformers)**: BERT utilizes the Transformer architecture and is pre-trained on large text corpora using objectives like Masked Language Modeling (predicting masked words) and Next Sentence Prediction. BERT produces contextualized embeddings by processing the entire input sequence simultaneously using self-attention. BERT embeddings are highly effective and are typically used by fine-tuning the pre-trained model on specific downstream tasks.

4.6 Evaluation of Word Embeddings

Word embeddings are evaluated using both intrinsic and extrinsic methods:

- **Intrinsic Evaluation**: Compares word embedding properties to human judgments or linguistic resources.
  - *Word Similarity Tasks*: Measures how well the cosine similarity between word vectors correlates with human-rated similarity scores for pairs of words.
  - *Analogy Tasks*: Assesses if the vector relationships capture analogical reasoning.
  - *Probing Tasks*: Specifically designed tasks to evaluate if embeddings encode particular linguistic features by training a simple classifier on top of the fixed embeddings.
- **Extrinsic Evaluation**: Evaluates the quality of embeddings by using them as features in a downstream NLP task and measuring the performance on that task.

4.7 Critiques and Bias in Embeddings

While powerful, word embeddings and the focus on large neural models have faced critiques.

- **Semantic Similarity vs. Meaning**: A key critique is that semantic similarity captured by vector proximity is only a "weak reflection" of actual linguistic meaning, which involves complex compositionality, pragmatics, and world knowledge.
- **Bias in Embeddings**: Word embeddings learn associations from the training data, and unfortunately, corpora often contain societal biases. This means embeddings can encode and perpetuate stereotypes.
- **Explainability**: It can be challenging to interpret what specific dimensions or combinations of dimensions in a dense vector space represent linguistically.

Chapter 5

Part of Speech (POS)

Words can be categorized into different classes based on their syntactic and morphological behavior, and the task of automatically identifying them is called Part of Speech Tagging.

5.1 Grammatical Categories of Words

Parts of Speech (POS), or word classes, represent grammatical categories of words.

5.1.1 The Significance of Parts of Speech

Many words in a language are ambiguous; they can belong to more than one category and thus have different meanings or functions depending on the context.

Consider the simple sentence: ”They can fish”.

- If ”can” is a modal verb (MD) and ”fish” is a base form verb (VB), the sentence means ”They have the ability to fish.” (They /PRP can /MD fish /VB).
- If ”can” is a non-third person singular present verb (VBP, meaning ’to put something in a can’) and ”fish” is a plural or mass noun (NN), the sentence means ”They preserve fish in cans.” (They /PRP can /VBP fish /NN).

This example clearly illustrates how the grammatical category assigned to each word (its POS tag) determines the intended meaning of the sentence.

5.1.2 Open vs. Closed Classes

Words in a language can broadly be divided into two main types of categories:

- **Closed Classes**: These are word categories that have a relatively fixed number of members. New words are rarely added to closed classes. They typically consist of function words that play important grammatical roles in a sentence. Words in closed classes tend to be short and occur very frequently.
- **Open Classes**: These are word categories that constantly acquire new members as the language evolves. These classes are virtually infinite.

5.1.3 Major Part of Speech Categories

The major categories of tagsets typically include:

- **Nouns (NN, NNS, NNP, NNPS)**: Words representing people, places, things, or ideas.
  - Common Nouns (NN, NNS): Refer to general entities (e.g., 'cat', 'money', 'snow'). Can be singular (NN) or plural (NNS).
  - Proper Nouns (NNP, NNPS): Refer to specific entities and are usually capitalized (e.g., 'Mary', 'Washington', 'Italy'). Can be singular (NNP) or plural (NNPS).
  - Count Nouns: Refer to entities that can be counted (e.g., 'day', 'child', 'Washington').
  - Mass Nouns: Refer to entities that cannot typically be counted as individual units (e.g., 'snow', 'salt', 'cutlery', 'mathematics'). Some words can be both count and mass depending on context.
  - Compound Nouns: Formed by combining two or more words, functioning as a single noun (e.g., 'New England Journal of Medicine', where the entire phrase can act as a Proper Noun).
- **Verbs (VB, VBD, VBG, VBN, VBP, VBZ)**: Words that refer to actions, processes, or states of being. Note that some languages may not have a clear noun/verb distinction.
- **Adjectives (JJ, JJR, JJS)**: Words that describe properties or qualities of nouns. Some languages, like Korean, may not have a distinct class of adjectives, using verbal constructions instead ("to be beautiful").
- **Adverbs (RB, RBR, RBS)**: A heterogeneous class of words that modify verbs, adjectives, other adverbs, or entire clauses. They can indicate manner, degree, time, location, etc.
- **Interjections (UH)**: Words or short phrases that express emotion or surprise (e.g., 'Oh', 'hey', 'alas', 'uh', 'um'). Includes greetings and simple responses ('hello', 'yes', 'no').
- **Conjunctions (CC, IN)**: Words that connect sentences, clauses, phrases, or words.
- **Pronouns (PRP, PRP$, WP, WP$)**: Words used as shorthand to refer to nouns or noun phrases, aiding communication efficiency.
- **Auxiliary Verbs (MD, VB, VBP, VBZ, VBG, VBN - when used as auxiliary)**: Verbs that accompany a main verb to express grammatical features such as tense, aspect (whether an action is completed or ongoing), and mood (necessity, possibility, desire).

These categories provide a framework for analyzing sentence structure.

5.2 Part of Speech Tagging Task

Part of Speech Tagging is the task of assigning the correct POS tag to each word in a given text (typically a sentence). It takes a sequence of words as input and produces a sequence of corresponding POS tags as output.

$$\text{Input: } w_1, w_2, w_3, \ldots, w_N$$

$$\text{Output: } t_1, t_2, t_3, \ldots, t_N$$

where w_i is the i-th word and t_i is its assigned POS tag.

This is considered a **sequence labeling task**, a common problem type in NLP where each element in a sequence is assigned a label. The primary challenge in POS tagging is **disambiguation**, as a significant portion of words (often over 50%) can have multiple possible POS tags. The correct tag must be chosen based on the word's context within the sentence.

Standard tagsets, like the Penn Treebank tagset, provide a defined set of tags to be used for annotation. For example, tagging the sentence "Mary will support the bill" would yield:

Mary /NNP will /MD support /VB the /DT bill /NN

5.3 Evaluation

The performance of a POS tagger is typically evaluated using **accuracy**, which is the percentage of words in a test set that are assigned the correct tag when compared to a human-annotated "gold standard" corpus.

Chapter 6

Named Entities Recognition (NER)

Named Entity Recognition (NER) aims to identify and classify mentions of rigid designators (like proper names) and certain specific terms in a text into predefined categories. NER is a key step in extracting structured information from unstructured text and is crucial for many downstream NLP applications.

Understanding a sentence often requires identifying the specific entities being discussed and their types. For example, in a user request like "Show me flights from Seattle to Boston next Monday," identifying "Seattle" and "Boston" as locations and "next Monday" as a time expression is essential for the system to correctly interpret the request and formulate a database query or perform an action.

Named entities are not just words; they refer to specific real-world concepts or instances. Once identified, named entities can be indexed, linked to knowledge bases, or related to other entities or information in the text.

6.1 Named Entity Types

The set of categories used in NER can vary depending on the task, domain, and annotation scheme. Different benchmarks and corpora have proposed different sets of entity types:

- **MUC-6 (Message Understanding Conference)**: An influential early standard that defined 7 entity types, including:
  - *Location*: regions, mountains, seas, cities.
  - *Person*: people, characters.
  - *Organization*: companies, organizational entities.
  - *Timex*: temporal expressions (dates, times of day).
  - *Numex*: numerical expressions (monetary values).
- **OntoNotes**: A large corpus with a more extensive annotation scheme, defining 18 entity types, including:
  - *PERSON*: People, including fictional.
  - *NORP*: Nationalities or religious or political groups.
  - *FACILITY*: Buildings, airports, highways, bridges, etc.
  - *ORGANIZATION*: Companies, agencies, institutions, etc.

- *GPE*: Countries, cities, states (Geo-Political Entity).
- *LOCATION*: Non-GPE locations, mountain ranges, bodies of water.
- *PRODUCT*: Vehicles, weapons, foods, etc. (Not services).
- *EVENT*: Named hurricanes, battles, wars, sports events, etc.
- *WORK OF ART*: Titles of books, songs, etc.
- *LAW*: Named documents made into laws.
- *LANGUAGE*: Any named language.
- *DATE*: Absolute or relative dates or periods.
- *TIME*: Times smaller than a day.
- *PERCENT*: Percentage (including "%"). *MONEY*: Monetary values, including unit.
- *QUANTITY*: Measurements, as of weight or distance. *ORDINAL*: "first", "second".
- *CARDINAL*: Numerals that do not fall under another type.

The specific set of relevant entities depends heavily on the domain and application (e.g., medical text requires recognizing diseases, treatments, symptoms).

6.2 Named Entity Recognition Task

NER is typically framed as a **sequence labeling task**. Given a sequence of tokens (words) in a sentence, the task is to assign a label to each token indicating whether it is part of a named entity and, if so, which type of entity it belongs to.

6.2.1 Sequence Coding Schemes

To represent the boundaries and types of named entities for sequence labeling models, specific coding schemes are used:

- **IO Coding**: A simple scheme with $N + 1$ tags, where N is the number of entity types. Tags are "I-EntityType" for tokens *inside* an entity span and "O" (Outside) for tokens not part of any entity. This scheme doesn't explicitly mark the beginning of an entity.
- **BIO Coding**: A more common scheme with $2N + 1$ tags. Tags are "B-EntityType" for the *beginning* token of an entity span, "I-EntityType" for *inside* tokens (subsequent tokens within the same entity span), and "O" for tokens outside any entity. This scheme explicitly marks entity boundaries, which is important for distinguishing adjacent entities of the same type. BIO coding is often equivalent to bracketed notation used in linguistic annotation.

The NER task involves two main steps: **Segmentation** (identifying the boundaries of the named entity spans) and **Labeling** (classifying the identified spans into the correct entity types). The primary challenge is **disambiguation**, as words or sequences of words can be ambiguous and refer to different entity types or not be an entity at all depending on context.

6.2.2 Nested Named Entities

A significant challenge in NER is identifying **nested named entities**, where one named entity is embedded within another, potentially of a different type. For example, "[the burial site of [Sheikh Abbad] PERSON] LOCATION" or "[[Cambridge] LOCATION University] ORG". Traditional linear sequence labeling models struggle with nested entities as they typically assign only one tag per token, often assigning the tag of the outermost entity. Recognizing nested entities is crucial for fully understanding complex phrases and is important for downstream applications like human-machine dialogue, where a failure to recognize nesting can lead to misunderstandings.

6.2.3 Open and Domain-Specific Named Entities

While common entity types like Person, Organization, and Location often exhibit strong naming conventions, entities in specific domains (e.g., biology, medicine) or "open" entity types (e.g., movie titles, song names, book titles), which can sometimes be long and lack structural regularity, pose additional challenges. High-performing NER in specialized or open domains often requires domain-specific data collection and annotation.

6.3 Evaluation Metrics

Evaluating the performance of NER systems requires metrics that account for both correct segmentation and correct labeling. The most common metrics are based on comparing the system's output to a gold standard annotation:

- **Precision**: Measures the proportion of entities identified by the system that are correct (both boundary and type match the gold standard).

$$\text{Precision} = \frac{\text{Number of Correctly Identified Entities}}{\text{Number of Entities Identified by System}}$$

- **Recall**: Measures the proportion of entities in the gold standard that were correctly identified by the system.

$$\text{Recall} = \frac{\text{Number of Correctly Identified Entities}}{\text{Number of Entities in Gold Standard}}$$

- **F1-Score**: The harmonic mean of precision and recall, providing a single score that balances both aspects. It is the standard metric for NER evaluation.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluation metrics can be sensitive to boundary errors (e.g., identifying "Bank of Chicago" instead of "First Bank of Chicago" might count as both a False Positive and a False Negative). Performance can also vary significantly depending on the text genre and the specific entity types being evaluated.

6.4 Challenges

NER continues to face several challenges:

- **Ambiguity**: Disambiguating entities based on context remains difficult.
- **Domain Adaptation**: Models trained on general data often perform poorly on specialized domains due to differences in vocabulary and entity types.
- **Multilingual NER**: Building NER systems for many languages, especially low-resource languages with limited annotated data, is challenging.
- **Handling Novel and Rare Entities**: Identifying entities that were not seen in training data.
- **Nested and Coreference Issues**: Accurately identifying nested entities and resolving coreference (when different text mentions refer to the same real-world entity).

Chapter 7

Sentence-Level Phenomena in NLP

Beyond identifying individual words, their categories (POS), or specific entities (NER), understanding natural language requires grasping the meaning and function of entire sentences. Sentences are the primary units through which speakers express propositions, ask questions, issue commands, and convey complex information.

7.1 The Role of Sentences in Communication

When humans communicate, they generate sequences of linguistic symbols (words) that belong to a shared code – natural language. These sequences of words, forming sentences and utterances, serve to:

- Embed the speaker's intentions (e.g., requesting information, providing an answer, expressing an opinion).
- Embed expected effects on the hearer (e.g., eliciting a response, causing a belief change, prompting an action).

This view emphasizes that sentences are not merely strings of words but carriers of meaning and communicative function.

7.2 Defining and Representing Sentence Meaning

A core task in NLU is mapping a natural language sentence into such a semantic representation. This representation often needs to capture not only the entities mentioned but also their relations, the overall state of affairs described, and the speaker's attitude or intent. This mapping can involve various techniques, including sequence labeling (mapping words or spans to semantic concepts or roles), semantic parsing (generating logical forms or graph representations), or frame semantic parsing (identifying semantic frames and their participants).

For example, the sentence "Show me morning flights from Boston to SF on Tuesday" can be mapped to a structured representation that identifies the user's goal (SHOW), the topic (FLIGHTS), the time (morning, Tuesday), and the origin/destination (Boston, SF). This mapping process bridges the gap between natural language and a form that computational systems can act upon.

7.3 Key Sentence-Level Linguistic Phenomena

Understanding the full meaning of a sentence requires analyzing various linguistic phenomena that go beyond the identification of individual words or simple entity spans.

7.3.1 Negation

Negation is a fundamental linguistic phenomenon that reverses the truth value of a proposition. Negation significantly impacts sentence meaning. NLU models must accurately detect the scope of negation and understand its semantic effect.

7.3.2 Modality

Modality affects the strength of assertions and inferences that can be drawn. ”She must be home” implies a higher degree of certainty than ”She might be home”. Interpreting modality is essential for understanding nuance, stance, and inferential meaning.

7.3.3 Presupposition

Presuppositions are underlying assumptions that are taken for granted in an utterance. They are triggered by specific linguistic constructions, verbs (e.g., ”stop”, ”regret”, ”know”), adverbs (e.g., ”again”, ”too”), or syntactic structures. Identifying presuppositions is crucial for a complete understanding of implied meaning.

7.3.4 Textual Entailment

Textual entailment is the directional relationship between two text fragments, where the truth of one text (the premise) logically implies the truth of the other (the hypothesis). Recognizing entailment is a core problem related to understanding inference and is important for tasks like question answering (finding text that entails the answer) and summarization (ensuring the summary is entailed by the original text).

7.3.5 Quantifiers and Generalized Quantifier Theory

Quantifiers express quantities or proportions and are crucial for understanding the scope of statements.

7.3.6 Scope Ambiguities

Scope ambiguity arises when a sentence contains multiple linguistic operators (like negation, quantifiers, or modals) and their hierarchical relationship is unclear, leading to multiple possible interpretations. Resolving scope ambiguities is necessary to determine the precise logical meaning of a sentence.

7.3.7 Focus and Information Structure

Focus refers to the part of a sentence that conveys new, important, or contrastive information, while information structure deals with how information is organized relative to the presumed knowledge of the listener.

7.3.8 Implicature

Implicatures are inferences that are suggested by an utterance in a particular context, even though they are not explicitly stated or logically entailed. Implicatures require pragmatic reasoning and are highly context-dependent, making them challenging for NLU systems.

7.3.9 Tense and Aspect

Tense relates the time of an event to the time of speaking. Aspect describes the internal temporal structure of an event or state, such as whether it is completed, ongoing, habitual, or instantaneous (e.g., simple, progressive, perfect).

7.3.10 Event and Situation Semantics

Formal semantic frameworks like Event Semantics and Situation Semantics provide ways to represent sentence meaning by explicitly modeling events, states, their participants (semantic roles), temporal properties, and the situations or contexts in which they occur. These approaches offer richer representations than simple predicate-argument structures.

7.3.11 Sentiment and Subjectivity

Analyzing sentiment and subjectivity involves identifying expressions of private states such as opinions, beliefs, emotions, and evaluations. Sentiment analysis, in particular, focuses on determining the positive, negative, or neutral tone of a text. Sentiment and subjective expressions are influenced by sentence-level phenomena like negation, modality, intensifiers, and also by the context in which they appear. Handling these complexities is vital for tasks like opinion mining, review analysis, and understanding subjective language.

7.4 Practical Representation: Intents and Slots

In practical conversational AI systems and NLU platforms (like those powering virtual assistants), sentence-level understanding is often simplified using the concepts of **Intents** and **Slots**.
An Intent represents the overall goal or action the user wants to perform .

Slots represent the specific pieces of information or parameters needed to fulfill that intent. The NLU component maps the user's utterance to a specific intent and extracts the relevant slot values. For example, "Book a flight to London tomorrow" might be mapped to the 'BookFlight' intent with 'destination=London' and 'date=tomorrow'. While this intent-slot structure provides a practical way to map utterances to actionable representations for specific domains, it is often a simplification of the full semantic and pragmatic complexity of natural language and may lack the richness and reusability of more formal linguistic theories or representations like semantic parses or discourse acts.

Chapter 8

Neural Networks in NLU

8.1 Word Representations for Neural Networks

Neural NLU systems rely on representing words and text in a format that the network can process.

8.1.1 Word Embeddings

A foundational concept is the use of **word embeddings**. Instead of sparse, high-dimensional representations like one-hot vectors, words are represented as dense, lower-dimensional vectors in a continuous vector space. In this space, words with similar meanings or contexts are mapped to points that are close to each other. These vectors are typically learned from large text corpora using techniques like Word2Vec, GloVe, or FastText (static embeddings) or are derived from larger pre-trained models like BERT (contextualized embeddings). Word embeddings capture syntactic and semantic relationships and serve as the input layer for more complex neural architectures.

8.1.2 Tokenization and Positional Encodings

Before creating embeddings, raw text must be converted into sequences of tokens. **Tokenization** splits text into units. For some modern neural networks like Transformers, sub-token level tokenization, such as **Byte-Pair Encoding (BPE)**, is used to handle large vocabularies and out-of-vocabulary words.

For models that process sequences without inherent recurrence (like Transformers), **Positional Encodings** are added to the word (or token) embeddings. These are additive embeddings that provide information about the position or order of tokens in the sequence, which is necessary for the model to utilize the sequential nature of language.

8.2 Neural Network Architectures for NLU

Different neural network architectures are designed to handle the specific characteristics of language data.

8.2.1 Feedforward Neural Networks (FNNs)

Feedforward Neural Networks are the simplest type, with information flowing in only one direction (input to output). Their main utility in NLU is for tasks that can be framed as simple classification or regression on context-independent features, or as components within more complex architectures.

8.2.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed to process sequences by maintaining a hidden state (also called context layer or state) that is updated at each time step as the network processes the sequence. This recurrent connection allows information from previous steps to influence the current step, enabling the model to have a form of memory and capture dependencies across time.

However, standard RNNs struggle with the *vanishing gradient problem*, which makes it difficult for them to learn and retain information over long sequences, limiting their ability to capture long-range dependencies. To address this, more sophisticated gated RNN variants were developed.

- **Long Short-Term Memory (LSTM) Networks**: LSTMs use gating mechanisms (input, forget, and output gates) to control the flow of information into and out of a memory cell, allowing them to effectively learn and remember information over long sequences and mitigate the vanishing gradient problem.
- **Gated Recurrent Units (GRUs)**: GRUs are a simpler variant of LSTMs, using fewer gates (reset and update gates) while achieving comparable performance on many tasks.

RNNs (including LSTMs and GRUs) were a significant step up from FNNs for sequence modeling, as their history representation is not limited by a fixed window size.

8.2.3 Bidirectional RNNs (BiRNNs)

Standard RNNs process sequences in only one direction (e.g., left-to-right). **Bidirectional RNNs (BiRNNs)** enhance this by processing the sequence in both forward and backward directions simultaneously. The hidden state at each time step is concatenated from the hidden states of the forward and backward passes. This allows the model to incorporate context from both the preceding and succeeding words, which is particularly beneficial for sequence labeling tasks like Part-of-Speech tagging or Named Entity Recognition where the label of a word can depend on words that come after it.

8.2.4 Transformer Models

The **Transformer** architecture, introduced in 2017, has revolutionized NLP and is the foundation for most modern state-of-the-art models, including Large Language Models (LLMs). Transformers move away from recurrent connections and rely entirely on **Self-Attention Mechanisms**.

- **Self-Attention**: This mechanism allows each token in a sequence to weigh the importance of all other tokens in the same sequence when computing its representation.
- **Multi-Head Attention**: The Transformer uses multiple attention heads in parallel to capture different types of relationships.

Transformers process tokens in parallel, making them more computationally efficient for training on large datasets compared to RNNs. Key Transformer-based models include:

- **BERT (Bidirectional Encoder Representations from Transformers)**: An encoder-only Transformer model pre-trained using Masked Language Modeling (predicting masked tokens) and Next Sentence Prediction objectives. BERT is designed for fine-tuning on various downstream tasks.
- **GPT (Generative Pre-trained Transformer)**: A decoder-only Transformer model pre-trained on a standard language modeling objective (predicting the next token). GPT models are primarily used for text generation.
- **T5 (Text-To-Text Transfer Transformer)**: An encoder-decoder Transformer model that frames all NLP tasks as text-to-text problems.

8.3 Intent Classification and Slot Filling

A common application of neural networks in NLU, particularly in conversational AI, is the joint task of **Intent Classification** and **Slot Filling**. Intent Classification is a text classification task where the model predicts the user's goal (Intent) from the input utterance. Slot Filling is a sequence labeling task (similar to NER) where the model identifies and labels the words or phrases (Slots) that provide the necessary parameters for the detected intent.

These two tasks are often inter-dependent; the required slots depend on the user's intent, and identifying certain slots can help determine the intent. Neural networks are well-suited for modeling this relationship.

8.3.1 Multi-task Learning

Multi-task Learning (MTL) is a paradigm where a single model is trained simultaneously on multiple related tasks, such as Intent Classification and Slot Filling. Models like Slot-Gated Modeling (based on RNNs) and joint BERT models have been developed for this purpose.

8.3.2 Two-Step Prediction Methods

Alternatively, Intent Classification and Slot Filling can be approached using two-step prediction methods:

- **Pipeline**: The output of one task (e.g., Intent Classification) is used as input features for the other task (e.g., Slot Filling).
- **Stack-propagation**: A method where the prediction of one task (e.g., Intent) is used in a differentiable way to inform the learning for the other task (e.g., Slots).

- **Two-Step with Shared Layers**: Models that share lower layers for initial processing but have separate layers for the final prediction of each task, potentially with co-interactive mechanisms between the tasks.

Benchmark datasets like ATIS and SNIPS are commonly used for training and evaluating models on the Intent Classification and Slot Filling tasks.

Chapter 9

Constituency and Dependency Grammars

9.1 Ambiguity and Parsing

A significant challenge in interpreting natural language sentences is ambiguity, where a sentence can have multiple possible structural interpretations, leading to different meanings.

While modern language models (LLMs) like GPT can sometimes disambiguate such sentences based on the vast patterns learned from data, they often rely heavily on inference rather than an inherent understanding of the underlying grammatical structure. This highlights the need for models that can explicitly represent and reason about sentence structure.

Parsing is the computational process of taking a string of words and a grammar and returning one or more parse trees that represent the possible syntactic structures of that string. Parsing is a crucial step towards building interpretable NLU systems.

9.2 Syntactic and Semantic Parsing

Natural Language Understanding often involves two interconnected types of parsing:

- **Syntactic Parsing**: This focuses on identifying the grammatical structure of a sentence, such as the boundaries of phrases (constituents) and the grammatical relationships between words.
- **Semantic Parsing**: This aims to map a natural language sentence into a formal representation of its meaning, determining elements like "who did what to whom".

9.3 Grammars and their Role

Grammars are formal systems that provide the tools to describe the relations among symbolic representations of language (such as words or sub-words). They define the rules that govern how words can be combined to form grammatical sentences, constraining the relative position and grouping of words.

The main objectives of grammars in NLP are the **analysis** of existing sentences to determine their structure and the **generation** of new grammatical sentences.

9.4 Constituency Grammars (Context-Free Grammars)

Constituency grammars, typically formalized as Context-Free Grammars (CFGs), model sentences by grouping words into hierarchical constituents or phrases. They capture both **constituency** (how words combine into units like Noun Phrases or Verb Phrases) and **ordering** (the rules governing the sequence of these units).

9.4.1 Defining CFGs Formally

Formally, a Context-Free Grammar G is defined as a 4-tuple (N, Σ, R, S), where:

- N is a finite set of **non-terminal symbols** (or variables), representing abstract grammatical categories (e.g., S, NP, VP).
- Σ is a finite set of **terminal symbols**, which are the words of the language (disjoint from N).
- R is a finite set of **production rules**, each of the form $A \rightarrow \alpha$, where A is a non-terminal from N and α is a string of zero or more symbols from (N ∪ Σ). Rules where α consists only of terminal symbols are often called lexicalization rules.
- S is a designated **start symbol** from N, typically 'S' for Sentence.

The term "context-free" means that a non-terminal on the left-hand side of a rule can be rewritten regardless of the context in which it appears; the rule $A \rightarrow \alpha$ can be applied whenever the symbol A is encountered.

9.4.2 Constituency and Ordering in English

CFGs capture the structure of key constituents in English, such as:

- **Noun Phrases (NPs)**: Groups of words functioning as a noun.
- **Verb Phrases (VPs)**: Groups of words containing a verb and its complements/modifiers. A basic rule is VP → Verb NP.
- **Prepositional Phrases (PPs)**: Groups of words starting with a preposition. A typical rule is PP → Preposition NP.
- **Sentences (S)**: The top-level constituent, often represented by the rule S → NP VP.

9.4.3 Recursion in CFGs

A powerful feature of CFGs is recursion, where a non-terminal can appear on both the left and right sides of a production rule. This allows grammars to generate or analyze sentences of arbitrary length and complexity. Examples include rules for attaching PPs to NPs or VPs:

- NP → NP PP: Allows structures like [[The flight] [to Boston]].

- VP $\rightarrow$ VP PP: Allows structures like [[departed Miami] [at noon]].

The recursive nature of these rules enables the representation of complex nested structures, such as a sequence of prepositional phrases modifying a noun, as seen in phrases like "Flights from Denver to Miami in February on a Friday". Each PP attaches to the preceding NP, creating nested brackets representing the constituency structure.

9.4.4 Derivations and Parse Trees

A **derivation** is a sequence of rule applications starting from the start symbol S and leading to a string of terminal symbols.

Derivations can be visually represented as **parse trees** (or constituency trees), where the root is the start symbol, internal nodes are non-terminals, and the leaves are the terminal symbols (words). The tree structure explicitly shows the hierarchical grouping of words into constituents. CFGs are both generative (they can produce grammatical strings) and analytical (they can impose a tree structure on a given string).

9.4.5 Challenges in CFGs

Despite their ability to model many syntactic structures, CFGs face challenges in capturing certain linguistic phenomena:

- **Agreement**: CFG rules like S $\rightarrow$ NP VP do not inherently enforce agreement features like number between the subject NP and the VP.
- **Subcategorization**: Verbs have specific requirements for the types and number of arguments they take (their complements). For example, "sneeze" is intransitive and doesn't take a direct object. Verb subcategorization information is crucial for accurate semantic analysis, helping to determine argument structure and "who did what to whom".
- **Movement**: Phenomena like those in Wh-questions ("Which flight do you want me to have the travel agent book?") involve constituents being moved from their expected positions, creating long-distance dependencies. The direct object "Which flight" is the object of "book" but appears at the beginning of the sentence, separated by other verbs.

9.4.6 Sentence Types and Coordination

CFGs can model various basic sentence types based on their typical structures:

- **Declaratives**: Statements (e.g., "I want a flight") often modeled by S $\rightarrow$ NP VP.
- **Imperatives**: Commands (e.g., "Leave!") often modeled by S $\rightarrow$ VP.
- **Yes-No Questions**: Questions answerable with yes/no (e.g., "Did the plane leave?").
- **WH-Questions**: Questions using Wh-words (e.g., "Which flights do you have?").

Coordination, joining similar constituents with conjunctions like "and," can be handled with recursive rules like X $\rightarrow$ X and X, where X is a variable representing any constituent type (S, NP, VP, etc.).

9.5 Dependency Grammars

In contrast to constituency grammars that focus on phrases as primary units, **Dependency Grammars** analyze the syntactic structure of a sentence based on binary, directed relationships between individual words. Each relationship connects a **head** (governor) word to a **dependent** (modifier) word.

Dependency structures are commonly visualized as trees where each word is a node, and a directed edge represents a dependency relation between a head and its dependent. Every word in the sentence typically has exactly one incoming dependency arc, except for the main verb or root of the sentence, which has none.

9.5.1 Typed Dependencies

Dependency relations are often **typed** with grammatical functions that specify the nature of the relationship between the head and the dependent. Common dependency types include:

- NSUBJ: Nominal Subject
- OBJ: Direct Object
- DET: Determiner
- COMPOUND: Part of a multiword unit

These typed dependencies provide explicit information about the grammatical roles of words.

Dependency grammars can be particularly well-suited for languages with relatively free word order, as the dependencies between words can be captured regardless of their linear arrangement.

9.6 Projectivity and Non-Projective Structures

A dependency tree is considered **projective** if, when the words are written in their linear order, the dependency arcs drawn above the words do not cross. Handling non-projectivity can be a challenge for some parsing algorithms.

9.7 Applications in Semantic Parsing

Dependency parsing facilitates semantic interpretation by explicitly representing the relationships between predicates (often verbs or nouns) and their arguments. This makes it easier to extract information about "who did what to whom" or identify relationships between entities.

Chapter 10

Meaning Representation in Natural Language Understanding

Meaning is conveyed through explicit signals such as speech, text, and gestures. This meaning needs to be represented in an abstract form that can be used by a machine for internal computations (e.g., inference) or for explaining machine actions and responses.

Testing for meaning understanding is not always straightforward. For example, answering "What is the capital of Italy?" with "Rome" might not necessarily mean the system understands the concepts of "capital", "Italy", "Rome", or their relation. Neural models, for instance, can provide correct answers without necessarily building an explicit intermediate abstract representation.

10.1 Word Meaning

Traditionally, meaning in natural language is studied from three perspectives:

- **Lexical Semantics**: The meanings of individual words.
- **Sentential Semantics**: How word meanings combine to form the meanings of sentences or utterances.
- **Discourse or Pragmatics**: How sentence meanings combine with each other and context (linguistic and non-linguistic) to form the meaning of a document or dialogue. This includes context from visual stimuli.

10.1.1 Lexemes, Lemmas, and Wordforms

- **Lexeme**: A unit of meaning denoted by a base form (orthographic or phonological).
- **Wordforms**: Different grammatical variants of a lexeme (e.g., "run", "runs", "ran", "running" are wordforms of the lexeme "run").
- **Lexicon**: A finite list of lexemes, similar to what is found in a thesaurus.
- **Lemma**: The citation form representing the lexeme (e.g., "Carpet" is the lemma for "Carpets").
- **Lemmatization**: The process of mapping wordforms to their lemma. This is not always deterministic due to multiple word senses (e.g., "found" can map

to "find" or "found" meaning 'to create an institution'). Lemmatization often requires knowledge of the word's category (POS) and its context.

- **Stemming**: A simpler process than lemmatization that involves stripping off suffixes (e.g., "going" $\rightarrow$ "go").

10.1.2 Relationships Between Word Meanings

Word meanings are related in various ways:

- **Homonymy**: Lexemes that share a form (phonological, orthographic, or both) but have unrelated, distinct meanings. Examples include "bank" (financial institution) vs. "bank" (riverside). Homonyms can be homophones (same sound, different spelling/meaning, e.g., "write" and "right") or homographs (same spelling, different sound/meaning, e.g., "bass" the fish vs. "bass" the instrument), or both.
- **Polysemy**: A single lexeme with multiple related meanings (e.g., "bank" the building vs. "bank" the financial institution). Most non-rare words are polysemous, and the number of meanings is often related to frequency. Verbs tend to be more polysemous. Distinguishing polysemy from homonymy can be difficult. Metaphor is considered an important type of polysemy. Metaphors can be difficult to interpret and compute if not explicitly coded in dictionaries.
- **Synonymy**: Words that have the same meaning in some or all contexts. Examples: "youth" / "adolescent", "big" / "large". A computational rule: two lexemes are synonyms if they can be substituted for each other in any sentence without changing the truth conditions. However, perfect synonyms are rare, as substitution may not preserve acceptability based on politeness, slang, register, genre, etc.. For example, "big sister" vs. "large sister".
- **Antonymy**: Words that are opposites with respect to one feature (scale) of their meaning, but otherwise very similar. Examples: "Dark" / "Light", "Hot" / "Cold".
- **Hypernymy/Hyponymy**: A hierarchical relationship where the meaning of one lexeme is a subset of the meaning of another. The more general term is the hypernym, and the more specific term is the hyponym. Example: "dog" is a hyponym of "canid", and "canid" is a hypernym of "dog". Similarly, "car" is a hyponym of "vehicle", and "vehicle" is a hypernym of "car". "Apple" is a hyponym of "fruit".

10.1.3 Lexical Resources

Lexical resources are collections of word forms, lexemes, and their relations.

- **WordNet**: One of the first examples (for English and European languages). It groups synonyms into sets called synsets, provides definitions and usage examples, and links synsets through relations like hypernymy/hyponymy. Building resources like WordNet requires intense manual effort and expertise.
- **ConceptNet**: A multilingual lexical resource.

10.2 Challenges in Natural Language Understanding

Several challenges exist in NLU:

- **Syntactic Ambiguity**: Sentences can have multiple possible parse trees.
- **Semantic Ambiguity**: Words or sentences can have multiple meanings (Homonymy, Polysemy). **Overspecification**: Providing too many details or conflicting constraints. **Underspecification**: Providing too few details or loose constraints.
- **NLU in Context**: Understanding language requires considering the linguistic and non-linguistic context.
- **Commonsense**: NLU systems need commonsense knowledge to understand implied actions or interpretations based on context (e.g., the implied actions to "open the door" vs. "open the bottle" vs. "open the radio", etc.).

10.3 Meaning Representations (MR)

Meaning representations are formal structures used to encode the meaning of natural language so that it can be processed and understood by machines.

10.3.1 Desiderata for Meaning Representations

Good Meaning Representations should have certain properties:

- **Verifiability**: The ability to connect the MR with a knowledge base of facts and relations. This is related to explainability. Example: "Is Kazakistan a Country?" $\rightarrow$ MR: IS-A(Kazakistan, Country).
- **Completeness (with respect to Ambiguity)**: For a given linguistic input, the MR should capture all plausible interpretations. Ambiguity can be resolved through downstream steps using world knowledge, dialogue context, or user feedback.
- **Robustness**: Multiple linguistic inputs with the same or similar meaning should map to the same canonical MR to improve knowledge base matching performance. Example: "Charrito has gluten-free dishes" and "You can get gluten free at Charrito" should map to MR: SERVES(Charrito, GlutenFreeFood).
- **Expressiveness**: The MR language should have sufficient coverage to represent the meanings of linguistic inputs. It should support inference (drawing conclusions from input and knowledge base) and handle variables (e.g., "a gluten-free restaurant" $\rightarrow$ MR: SERVES(x, GlutenFreeFood)).
- **Uncertainty**: The ability to evaluate the reliability or confidence of a target MR or its parts, based on the certainty of linguistic elements (syntax, lexemes, speech, etc.). Example: SERVES[0.85](Charrito[0.8], GlutenFreeFood[0.5]).

10.3.2 Representations of Meaning

Some of the most relevant types of Meaning Representations include:

- **Semantic Networks**: A network representing semantic relations between concepts or frames. These are directed/undirected graphs with vertices (concepts/frames) and edges (relations). Example for "I buy a stock": nodes for "Buying", "Buyer", "Speaker", "BoughtThing", "Stock" connected by edges representing the roles. Semantic networks, Frame-Based representations, and First Order Logic are, in principle, equivalent.
- **Frame-Based Representations**: Uses frames, which are structures with slots (features) and slot values (atomic values or embedded frames). Example for "I buy a stock": a "Buying" frame with slots like "Buyer" and "BoughtThing", where values are "Speaker" and a "Stock" frame respectively.
- **First Order Logic (FOL)**: An abstraction from natural language built around objects and relations. It can express facts about objects in a universe.
  - **Objects**: Entities in the world (e.g., houses, lake, one, two, three).
  - **Relations (or Properties)**: Unary or n-ary operators describing relationships between objects (e.g., IsBeautiful(houses), equals(one plus two, three)).
  - **Functions**: Special relations assigning only one output value for a given input (e.g., plus(one, two) returns three).
  - **Syntax Examples**:
    * "I buy a stock": $\exists e, y \text{ Buying}(e) \wedge \text{Buyer}(e, \text{Speaker}) \wedge \text{BoughtThing}(e, y) \wedge \text{Stock}(y)$ (There exists an event e of Buying, where the Buyer is the Speaker, the BoughtThing is y, and y is a Stock).
    * "Richard's brothers are John and Geoffrey": $\text{Brother(John, Richard)} \wedge \text{Brother(Geoffrey, Richard)} \wedge \text{John} \neq \text{Geoffrey} \wedge \forall x (\text{Brother}(x, \text{Richard}) \Rightarrow (x = \text{John} \vee x = \text{Geoffrey}))$ (John and Richard are brothers, Geoffrey and Richard are brothers, John and Geoffrey are distinct, and for all x, if x is a brother of Richard, then x is either John or Geoffrey).
  - **Simplified (Database Semantics)**: Often, the full expressiveness of FOL is simplified for practical purposes, assuming certainty about the identity of objects and facts. For "Richard's brothers are John and Geoffrey", this might be simplified to
    Brother(John, Richard) $\wedge$ Brother(Geoffrey, Richard).
    This simplification misses the uniqueness constraint (that John and Geoffrey are the *only* brothers). Database semantics simplifies encoding knowledge but assumes certainty, which may not always hold.
  - **Limitations of FOL**: FOL assumes the world consists of objects and relations that either hold or do not hold. This is restrictive and not always realistic for describing object relations, potentially requiring additional frameworks like probabilistic systems.

Chapter 11

Parsing Affective States - Sentiment Analysis

Sentiment Analysis or **Opinion Mining**, seeks to computationally identify, extract, quantify, and study affective information.

11.1 Why Parse Affective Meaning? Motivations and Applications

The drive to compute affective meaning stems from its utility in two broad categories: analysis/perception and synthesis/control.

11.1.1 Analysis and Perception

Understanding the emotional tone and sentiment in language allows us to:

- **Gauge public opinion:** Determine sentiment towards products, brands, political figures, or policies by analyzing social media, news articles, and reviews.
- **Enhance customer service:** Distill the emotions of callers to a helpline to better understand their needs and frustrations, potentially routing them to appropriate support or flagging urgent issues.
- **Monitor well-being and safety:** Detect stress in drivers or pilots from their speech, identify signs of depression or other medical conditions from written or spoken language, or recognize confusion in students interacting with e-tutoring systems.

11.1.2 Synthesis and Control

Beyond just understanding, computing affective meaning can enable systems to:

- **Create more engaging interactions:** Develop literacy tutors for children that can adapt to their emotional state, or design computer games with characters that respond realistically to player emotions.
- **Build empathetic dialogue systems:** Equip conversational AI to recognize a user's emotional state (e.g., frustration, happiness) and react appropriately, leading to more natural and effective human-computer interactions.

Real-World Application Domains

The practical applications of parsing affective states are vast and continue to grow:

- Media Analytics
- Opinion Tracking and Forecasting
- Multimedia Monitoring
- Education
- E-commerce
- and more...

11.2 Defining and Understanding Affective States

To parse affective states, we first need to understand what they are. The terminology can be complex, with "emotion" and "sentiment" often used interchangeably, though they have distinct nuances.

11.2.1 Emotion

Emotions serve to:

- **Explain behavioral discrepancies:** They help us understand why people's actions or feelings might deviate from what seems logical, appropriate, or useful in a given situation.
- **Account for individual differences:** They explain why different people react differently to the same event, or why the same person might react differently to the same situation on different occasions.

Emotion Arousal

Emotions are often triggered by specific situations or events. These emotional shifts are often accompanied by measurable physiological and behavioral changes, such as variations in speech characteristics like:

- Median Pitch: Tends to increase with arousal (e.g., higher in angry or frustrated states compared to neutral).
- Mean Energy (Loudness): Can also increase with heightened emotion.
- Speaking Rate: May increase or decrease depending on the specific emotion and individual.

Theories of Emotion

Two main families of theories are commonly referenced in computational approaches:

- **Basic or Discrete Emotions:** This perspective, championed by theorists like Tomkins, Plutchik, and Ekman, posits that there is a limited set of fundamental, universal emotions. Ekman, for example, identified six basic emotions widely recognized across cultures: **happiness, surprise, fear, sadness, anger, and disgust**. Plutchik proposed a "wheel of emotions" with eight basic emotions in four opposing pairs: joy-sadness, anger-fear, trust-disgust, and anticipation-surprise. These theories emphasize the universality of these core emotional experiences and their expressions (e.g., facial expressions).

- **Dimensional Emotions:** This approach, notably developed by Mehrabian and Russel, describes emotions not as discrete categories but as points in a continuous multi-dimensional space. The most common dimensions are:
  - **Valence:** The pleasantness or unpleasantness of the emotional experience (ranging from positive, like happy, to negative, like unhappy).
  - **Arousal:** The intensity of the emotion, or the level of physiological activation (ranging from excited or agitated to calm or sleepy).
  - **Dominance (or Control):** The degree to which an individual feels in control or influential versus controlled or submissive in the situation (e.g., anger might be high dominance, while fear might be low dominance).

  A simple two-dimensional model (Valence-Arousal) can map anger to high arousal/low pleasure, excitement to high arousal/high pleasure, sadness to low arousal/low pleasure, and relaxation to low arousal/high pleasure.

11.2.2 Sentiment

While closely related to emotion, **sentiment** is often conceptualized as a more enduring emotional attitude or disposition towards a particular object, person, or concept. It's distinct from a fleeting emotion triggered by an immediate situation.

Sentiment, therefore, often refers to a more stable, longer-term evaluation (positive, negative, or neutral) directed at an entity.

11.2.3 A Typology of Affective States

Klaus Scherer proposed a helpful typology distinguishing various affective phenomena:

- **Emotions:** Brief, organically synchronized evaluations of a major event or situation (e.g., angry, sad, joyful, fearful, ashamed, proud, elated). They are typically intense and short-lived.
- **Attitudes:** More enduring, affectively colored beliefs and dispositions towards specific objects or persons (e.g., liking, loving, hating, valuing, desiring). Sentiments fall under this category.

Other affective states include moods (longer-lasting, less intense, and often without a specific object) and interpersonal stances (affective stance taken towards another person in an interaction).

11.3 Language and Affect: The Role of Lexicons

A key component in computationally parsing affective states is understanding how affect is encoded in language, particularly at the word level. Words themselves can carry affective connotations beyond their literal meanings.

11.3.1 Sentiment Lexicons

A **lexicon**, in this context, is essentially a dictionary or list of words where each word is associated with some affective meaning or class. These associations can be categorical (e.g., positive, negative, angry, sad) or numeric (e.g., a score for valence, arousal, or intensity).

Sentiment lexicons are fundamental tools in affective computing. They can be used:

- As simple classifiers (e.g., counting positive vs. negative words in a text).
- As features for more sophisticated machine learning models.

Creation of Sentiment Lexicons

These valuable resources are created through various methods:

- **Human Annotation:**
  - **Psychologist-curated:** Experts in psychology or linguistics manually label words.
  - **Crowdsourced:** Large numbers of non-expert annotators (e.g., on platforms like Amazon Mechanical Turk - AMT) are asked to label words. This often involves specific experimental designs, like requiring multiple "Turkers" to annotate each item (Human Intelligence Task - HIT) to ensure reliability. Best-Worst Scaling is one such method where annotators choose the most and least associated word with an emotion from a set.
- **Semi-supervised or Supervised Machine Learning Models:** These approaches learn to predict the affective content of words based on labeled examples or by leveraging patterns in large text corpora.
- **Combination of Methods:** Many lexicons are built using a hybrid approach.

Examples of Notable Sentiment Lexicons

- **The General Inquirer (GI):** One of the earliest, developed by Philip Stone and colleagues. It primarily categorizes words based on binary valence (Positive: e.g., *admire, charm, excellent*; Negative: e.g., *anger, bad, deceit*).
- **MPQA Subjectivity Lexicon:** Developed by Theresa Wilson and colleagues, this lexicon contains lists of positive and negative words, with each entry also labeled for reliability (strongly subjective or weakly subjective).
- **Hu and Liu's Polarity Lexicon:** Created by Minqing Hu and Bing Liu, this lexicon was derived from product reviews and uses a bootstrapping method with WordNet to identify positive and negative words.
- **NRC Valence, Arousal, and Dominance (VAD) Lexicon:** Developed by Saif Mohammad, this lexicon assigns scores for valence, arousal, and dominance to a large set of English words (e.g., "vacation" has high valence, "enraged" has high arousal, "powerful" has high dominance).
- **NRC Word-Emotion Association Lexicon (EmoLex):** Also from Saif Mohammad and Peter Turney, EmoLex associates words with Plutchik's eight basic emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) as well as positive and negative sentiment. For example, "reward" is associated

with anticipation, joy, surprise, and trust.

11.4 Sentiment Analysis and Opinion Mining in Practice

Moving beyond individual words, sentiment analysis aims to understand the overall affective tone of larger text units—phrases, sentences, or entire documents. Opinion mining often focuses on a more granular level, seeking to identify specific components of an opinion.

11.4.1 The Structure of Opinions

An opinion can often be formally defined as a quadruple (or sometimes a quintuple):

- `g` **(Target or Topic):** The entity or aspect of an entity that the opinion is about.
- `s` **(Sentiment or Opinion):** The sentiment expressed towards the target (e.g., positive, negative, neutral, or a specific emotion).
- `h` **(Holder or Source):** The person or entity expressing the opinion.
- `t` **(Time):** The time when the opinion was expressed.

Sometimes, the **aspect** of the target is also explicitly identified (e.g., "The *screen* of this *phone* is *amazing*").

Consider a restaurant review on a platform like TripAdvisor. The review itself is a semi-structured document. The platform might pre-define aspects like "Food," "Service," "Value," and "Atmosphere," for which users can provide ratings. The free-text review then elaborates on these or other aspects.

Example: "The *pizza* (target/aspect) was *delicious* (sentiment: positive), but the *service* (target/aspect) was *terribly slow* (sentiment: negative)." Here, the reviewer (holder) expresses opinions at a certain time.

11.4.2 Challenges in Opinion Mining

- **Target and Aspect Identification:** Automatically identifying what the opinion is about can be difficult, especially when opinions are expressed implicitly or about multiple aspects.
- **Context Dependency – Words Are Not Alone:** The sentiment of a word can change drastically depending on the context and the domain.
  - The adjective "hot":
    - Restaurant domain: "hot pizza" (can be positive if it means freshly cooked, or negative if too spicy/temperature), "hot waiters" (likely positive).
    - Electronics domain: "hot CPU," "hot battery," "run hot" (almost always negative, indicating overheating).
  - The adjective "long":
    - Restaurant domain: "long queue," "long wait" (negative), "long menu," "long wine list" (could be positive or neutral).

        * Electronic products: "long battery life" (positive), "long delivery time" (negative).

  This highlights that simple lexicon lookups are often insufficient; deeper contextual understanding is needed.

11.4.3 Sentiment and Emotions Across a Document

The affective tone is not always static throughout a piece of text.

- In an email, chat message, or social media post, the sentiment can shift.
- Narratives often involve a progression of emotions. The same applies to conversations.

Understanding this **discourse context** is crucial. A negative statement might be mitigated by a subsequent positive one, or irony and sarcasm can flip the apparent polarity of words.

11.5 The Rise of Advanced Models

Recent advancements in NLU, particularly with large language models (LLMs) like GPT-3.5 and GPT-4, have shown remarkable capabilities in sentiment analysis and understanding nuanced affective expressions. These models, trained on vast amounts of text data, can often infer sentiment and emotion with high accuracy, even in complex contexts, without relying explicitly on traditional lexicons or rule-based systems. Their ability to capture subtle contextual cues represents a significant leap forward in the field. However, understanding their internal reasoning and ensuring their robustness and fairness remain active areas of research.