Introduction to Natural Language Processing

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

July 13, 2025

Natural Language Processing

Table of Contents

1. Motivation
2. Learning Outcomes
3. Introduction
4. N-grams
5. Sequence Notation
6. Probabilistic Notation
7. Count and Probability Matrices
8. Probability of a Sequence
9. Start and End Tokens
10. Out of vocabulary words
11. Limitations
12. Summary
13. References

Motivation

- Language helps us talk to each other—and now, to computers too!
- NLP (Natural Language Processing) teaches machines to understand, interpret, and generate human language.
- Why is NLP important?
  - Makes **chatbots** like ChatGPT and Siri possible
  - Powers **search engines** like Google
  - Helps **translate languages** (Google Translate)
  - Finds out what **people feel in reviews** (sentiment analysis)
- NLP is foundational to advanced AI systems.

Learning Outcomes

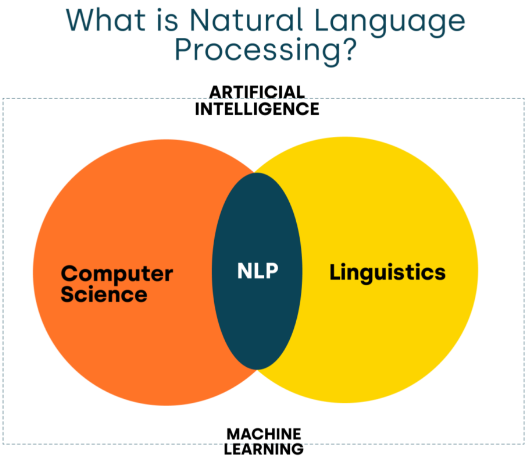
- **Define** Natural Language Processing (NLP)
- **Explain and construct** N-grams (unigram, bigram, trigram)
- **Understand** sequence notation and tokenization
- **Compute** N-gram probabilities and count matrices
- **Apply** start/end tokens and handle unknown words (OOV, UNK)
- **Recognize** the limitations of N-gram models and future directions

Natural Language Processing: **Introduction**

What is NLP?

- Study of computational approaches to processing natural languages.
- Processing includes:
  - Acquiring language data
  - Representing information
  - Storing text and speech
  - Understanding meaning
  - Characterizing language patterns
  - Generating new language
- Natural languages refer to human languages.

NLP - Introduction (cont.)



NLP = Computer Science + Linguistics + AI

- **Deals with:**
  - Language understanding (input)
  - Language generation (output)
  - Language translation
  - Information extraction
- **Subfields:**
  - Syntax
  - Semantics
  - Pragmatics
  - Discourse

Goal

Goal: Deep Understanding
Requires context, linguistic structure, meanings...

To Avoid: Shallow Matching
Could be useful also though depending on use case

Goal of NLP:

- Enable machines to understand and generate human language.
- Facilitate human-computer interaction through natural language.
- Develop systems that can process and analyze large amounts of text data.

NLP Pipeline

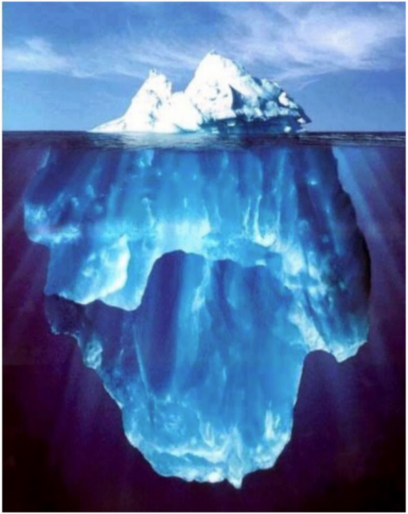
- **Text Preprocessing**
  - Cleaning and preparing raw text for analysis.
- **Tokenization**
  - Splitting text into words, sentences, or other meaningful units.
- **POS Tagging**
  - Assigning parts of speech (noun, verb, etc.) to each token.
- **Parsing**
  - Analyzing grammatical structure of sentences.
- **Named Entity Recognition (NER)**
  - Identifying entities such as people, organizations, locations.
- **Sentiment Analysis / Classification**
  - Determining sentiment or categorizing text.
- **Language Modeling**
  - Predicting the next word or sequence in text.

Text Data is Superficial

- An iceberg is a large piece of freshwater ice that has broken off from a snow-formed glacier or ice shelf and is floating in open water.

Text Data is Superficial

- An iceberg is a large piece of freshwater ice that has broken off from a snow-formed glacier or ice shelf and is floating in open water.



NLP Basics – What It's All About

Teaching machines to **understand and generate human language**

- **Text preprocessing**:
  Cleaning and preparing raw text data (removing noise, tokenization, normalization).
- **Understanding meaning**:
  Extracting meaning from text using techniques like part-of-speech tagging, named entity recognition, and sentiment analysis.
- **Language modeling**:
  Building models that can predict or generate text, such as autocomplete or next-word prediction.
- **Translation, summarization, etc.**:
  Enabling applications like machine translation, text summarization, question answering, and more.

Text data is everywhere: tweets, reviews, articles, chats! NLP helps us make sense of this vast information.

Core NLP Tasks

Task	What It Does	Example
Tokenization	Split text into words	`"I love NLP"` → `["I", "love", "NLP"]`
POS Tagging	Label grammar tags	`"Dogs bark"` → [Noun, Verb]
Named Entity Recognition	Find names, places, etc.	`"Christopher Nolan lives in Los Angeles"`
Sentiment Analysis	Detect mood	`"This movie was amazing!"` → Positive
Machine Translation	Language to language	English → French

Text Preprocessing

- **Lowercase everything**:
  Example: `"NLP"` → `"nlp"`
- **Removing stop words**:
  Eliminate common words that carry little meaning (e.g., "is", "the", "and") to focus on important content.
- **Stemming/Lemmatization**:
  Reduce words to their root or base form.
  Example: `"running"` → `"run"`
- **Vectorization**:
  Transform words or documents into numerical representations for machine learning models:
  - **Bag of Words**: Counts word occurrences in a document.
  - **TF-IDF (Term Frequency-Inverse Document Frequency)**: Weighs words by importance across documents.
  - **Word2Vec**: Learns dense vector representations capturing word meaning and context.

What is a corpus?

- A corpus is a collection of text.
- Often annotated in some way.
- Sometimes just lots of text.
- **Balanced corpora:** Usually not possible in practice.
- **Examples:**
  - Newswire collections: 500M+ words
  - Brown corpus: 1M words of tagged "balanced" text
  - Penn Treebank: 1M words of parsed WSJ
  - Canadian Hansards: 10M+ words of aligned French/English sentences
  - The Web: billions of words of who knows what

What is Vocabulary in NLP?

- **Vocabulary** = All unique words in your dataset
- Example: `"I love NLP and NLP loves me"` → Vocabulary = {`"I"`, `"love"`, `"NLP"`, `"and"`, `"loves"`, `"me"`}
- More data = Bigger vocabulary = Harder to process!

Tip: Rare words may not help; common words may not mean much.

The Problem with Sparse Representations

Traditional approach: One-hot encoding

- Example: `"NLP"` $\rightarrow [0, 0, 1, 0, 0, 0, 0 \ldots]$

Problem:

- High-dimensional (thousands of words!)
- Sparse (mostly 0s)
- No meaning in structure (no relation between `"king"` and `"queen"`)
- Not efficient for learning

Word Frequencies & Feature Extraction

Feature extraction = Turning text into numbers

- Count how often each word appears (**Term Frequency**)

Example:

Text	"great product"	"bad product"
Word: "great"	1	0
Word: "bad"	0	1

Use this to find patterns in sentiment, spam, etc.

Positive & Negative Frequencies

Suppose you're classifying reviews:

Positive Reviews: `["amazing", "good", "great"]`
Negative Reviews: `["bad", "awful", "terrible"]`

Count how often each word appears in each class.

Example table:

Word	**Positive Count**	**Negative Count**
good	20	1
bad	1	30

Helps models detect the "tone" (**sentiment clues**) of new text.

Feature Extraction Techniques

- **Bag of Words (BoW)**: Just counts word frequencies in each document.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Adjusts for how "unique" or important a word is in a document compared to all documents.
  - Words like `"the"`, `"is"` are less important.
- **Word Embeddings (later)**: Add meaning and capture relationships between words (e.g., similarity, analogy).

Natural Language Processing: **N-grams**

What are N-grams?

أكاديمية كاوست
KAUST ACADEMY

An N-gram is a sequence of N words

Corpus: I am happy because I am learning

Unigrams: { I , am , happy , because , learning }

Bigrams: { I am , am happy , happy because ... }

I happy

Trigrams: { I am happy , am happy because, ... }

- Items are typically **words** or **characters**.

Tokenization

- Tokenization is the process of breaking text into smaller units called **tokens**.
- Tokens can be words, characters, or subwords.
- Example: "I love NLP" can be tokenized into:
  - Words: ["I", "love", "NLP"]
  - Characters: ["I", " ", "l", "o", "v", "e", " ", "N", "L", "P"]
  - Subwords: ["I", " ", "lov", "e", " ", "N", "L", "P"]
- Tokenization is crucial for preparing text data for NLP tasks.

Examples of N-grams

Sentence: "I love NLP"

- **Unigrams:** I, love, NLP
- **Bigrams:** I love, love NLP
- **Trigrams:** I love NLP

Why Use N-grams?

- Capture local word co-occurrence
- Build simple language models
- Easy to compute and analyze
- Trade-off between simplicity (unigram) and contextual richness (trigram)

Sequence Notation Basics

Sentence: $w_1, w_2, \ldots, w_n$

For example: $w_1 = \text{I}, \; w_2 = \text{love}, \; w_3 = \text{NLP}$

General representation:

- Unigram: $P(w_i)$
- Bigram: $P(w_i \mid w_{i-1})$
- Trigram: $P(w_i \mid w_{i-2}, \; w_{i-1})$

Sequence Notation Example:

Corpus: $\underset{w_1 \quad w_2 \quad w_3}{\text{This is great}}$ ... $\underset{w_{498} \quad w_{499} \quad w_{500}}{\text{teacher drinks tea.}}$ $m = 500$

$$w_1^m = w_1 \; w_2 \; \dots \; w_m$$

$$w_1^3 = w_1 \; w_2 \; w_3$$

$$w_{m-2}^m = w_{m-2} \; w_{m-1} \; w_m$$

Sequence Notation Basics

Sentence: $w_1, w_2, \ldots, w_n$

For example: $w_1 = \text{I}, \;\; w_2 = \text{love}, \;\; w_3 = \text{NLP}$

General representation:

- Unigram: $P(w_i)$
- Bigram: $P(w_i \mid w_{i-1})$
- Trigram: $P(w_i \mid w_{i-2}, \; w_{i-1})$

Unigram probability

Corpus: $\boxed{\text{I}}$ am $\boxed{\text{happy}}$ because $\boxed{\text{I}}$ am learning

Size of corpus m = 7

$$P(I) = \frac{2}{7} \qquad P(happy) = \frac{1}{7}$$

Probability of unigram: $$P(w) = \frac{C(w)}{m}$$

Bigram probability

Corpus: I am happy because I am learning

$$P(am|I) = \frac{C(I\ am)}{C(I)} = \frac{2}{2} = 1$$

$$P(happy|I) = \frac{C(I\ happy)}{C(I)} = \frac{0}{2} = 0$$ I happy

$$P(learning|am) = \frac{C(am\ learning)}{C(am)} = \frac{1}{2}$$

Probability of a bigram: $$P(y|x) = \frac{C(x\ y)}{\sum_w C(x\ w)} = \frac{C(x\ y)}{C(x)}$$

Trigram probability

Corpus: I am happy because I am learning

$$P(happy|I\ am) = \frac{C(I\ am\ happy)}{C(I\ am)} = \frac{1}{2}$$

Probability of a trigram: $$P(w_3|w_1^2) = \frac{C(w_1^2\ w_3)}{C(w_1^2)}$$

$$C(w_1^2\ w_3) = C(w_1\ w_2\ w_3) = C(w_1^3)$$

N-gram probability

Probability of N-gram: $$P(w_N | w_1^{N-1}) = \frac{\boxed{C(w_1^{N-1} w_N)}}{C(w_1^{N-1})}$$

$$C(w_1^{N-1} w_N) = C(w_1^N)$$

Objective: Compute the probability of a sentence

N-gram Assumption: The probability of a word depends only on the previous $(n-1)$ words.

Formula:

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-n+1}^{i-1})$$

where w_1^n denotes the sequence $w_1, w_2, \ldots, w_n$ and w_{i-n+1}^{i-1} is the context of the previous $(n-1)$ words.

Bigram MLE:

$$P(w_i \mid w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

- $\text{Count}(w_{i-1}, w_i)$: Number of times the bigram (w_{i-1}, w_i) appears in the corpus.
- $\text{Count}(w_{i-1})$: Number of times the word w_{i-1} appears as a context.

Text: “I love NLP. I love AI.”

Bigrams and Counts:

- (I, love): 2
- (love, NLP): 1
- (love, AI): 1

Probability Calculations:

- $P(\text{love} \mid \text{I}) = \frac{2}{2} = 1.0$
- $P(\text{NLP} \mid \text{love}) = \frac{1}{2} = 0.5$
- $P(\text{AI} \mid \text{love}) = \frac{1}{2} = 0.5$

Count and Probability Matrices

Count Matrix: A matrix that counts occurrences of word pairs in a corpus.

Probability Matrix: A matrix that calculates probabilities of word pairs based on counts.

Example: For the sentence "I love NLP. I love AI."

- Count Matrix: Counts how many times each word appears with every other word.
- Probability Matrix: Calculates the probability of each word appearing given the previous word.

Count Matrix

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{\boxed{C(w_{n-N+1}^{n-1}, w_n)}}{C(w_{n-N+1}^{n-1})}$$

- Rows: unique corpus (N-1)-grams
- Columns: unique corpus words

Corpus: <s>I study I learn</s>

- Bigram count matrix

"study I" bigram

	<s>	</s>	I	study	learn
<s>	0	0	1	0	0
</s>	0	0	0	0	0
I	0	0	0	1	1
study	0	0	1	0	0
learn	0	1	0	0	0

Count Matrix Example

Text: "I love NLP. I love AI."

Bigrams:

- (I, love): 2
- (love, NLP): 1
- (love, AI): 1

Count Matrix:

- Rows: Words in the corpus
- Columns: Words in the corpus
- Cells: Count of occurrences of each word pair

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}, w_n)}{\boxed{C(w_{n-N+1}^{n-1})}}$$

- Divide each cell by its row sum

$$sum(row) = \sum_{w \in V} C(w_{n-N+1}^{n-1}, w) = C(w_{n-N+1}^{n-1})$$

Corpus: <s>I study I learn</s>

Count matrix (bigram)

	<s>	</s>	I	study	learn	sum
<s>	0	0	1	0	0	1
</s>	0	0	0	0	0	0
I	0	0	0	1	1	2
study	0	0	1	0	0	1
learn	0	1	0	0	0	1

Probability matrix

	<s>	</s>	I	study	learn
<s>	0	0	1	0	0
</s>	0	0	0	0	0
I	0	0	0	0.5	0.5
study	0	0	1	0	0
learn	0	1	0	0	0

Probability Matrix Example

Probability Calculation:

Bigram Probability:

$$P(\text{love} \mid \text{I}) = \frac{\text{Count}(\text{I}, \text{love})}{\text{Count}(\text{I})} = \frac{2}{2} = 1$$

Probability Matrix:

- Rows: Words in the corpus
- Columns: Words in the corpus
- Cells: Probability of each word given the previous word

Probability of a sequence

- Given a sentence, what is its probability?

$$P(the\ teacher\ drinks\ tea) = ?$$

- Conditional probability and chain rule reminder

$$P(B|A) = \frac{P(A,B)}{P(A)} \implies P(A,B) = P(A)P(B|A)$$

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

Probability of a sequence

$$P(the\ teacher\ drinks\ tea) =$$
$$P(the)P(teacher|the)P(drinks|the\ teacher)$$
$$P(tea|the\ teacher\ drinks)$$

Sentence not in corpus

Problem: Corpus almost never contains the exact sentence we're interested in or even its longer subsequences!

$$P(the\ teacher\ drinks\ tea) =$$

$$P(the)P(teacher|the)P(drinks|the\ teacher)$$
$$P(tea|the\ teacher\ drinks)$$

Approximation of sequence probability

the teacher drinks tea

$$P(tea|the\ teacher\ drinks) \approx P(tea|drinks)$$

$$P(teacher|the)$$
$$P(drinks|teacher)$$
$$P(tea|drinks)$$

$$P(the\ teacher\ drinks\ tea) =$$

$$P(the)P(teacher|the)P(drinks|the\ teacher)\boxed{P(tea|the\ teacher\ drinks)}$$

$$\Downarrow$$

$$P(the)P(teacher|the)P(drinks|teacher)\boxed{P(tea|drinks)}$$

Approximation of sequence probability

- Markov assumption: only last N words matter
- Bigram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$
- N-gram $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$
- Entire sentence modeled with bigram $P(w_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$

$$P(w_1^n) \approx \boxed{P(w_1)} P(w_2|w_1)...P(w_n|w_{n-1})$$

Quiz

Objective: Apply sequence probability approximation with bigrams.

Question:
Given these conditional probabilities
$P(Mary)=0.1$; $P(likes)=0.2$; $P(cats)=0.3$
$P(Mary|likes)=0.2$; $P(likes|Mary)=0.3$; $P(cats|likes)=0.1$; $P(likes|cats)=0.4$

Approximate the probability of the following sentence with bigrams: "Mary likes cats"

Type: Multiple Choice, single answer

Options and solution:

1. $P(\text{Mary likes cats}) = 0$
2. $P(\text{Mary likes cats}) = 1$
3. $\boxed{P(\text{Mary likes cats}) = 0.003}$
4. $P(\text{Mary likes cats}) = 0.008$

N-gram Models: **Start and End Tokens**

Why Use Special Tokens?

- **Start:** `<s>` or `<start>`
- **End:** `</s>` or `<end>`

Benefits:

- Helps model sentence boundaries
- Enables generation and evaluation

Start of sentence token <s>

the teacher drinks tea

$$P(the\ teacher\ drinks\ tea) \approx \boxed{P(the)} P(teacher|the)P(drinks|teacher)P(tea|drinks)$$

<s> the teacher drinks tea

$$P(<s>\ the\ teacher\ drinks\ tea) \approx \boxed{P(the|<s>)} P(teacher|the)P(drinks|teacher)P(tea|drinks)$$

- Trigram:

$$P(the\ teacher\ drinks\ tea) \approx \boxed{P(the)P(teacher|the)}P(drinks|the\ teacher)P(tea|teacher\ drinks)$$

the teacher drinks tea => <s> <s> the teacher drinks tea

$$P(w_1^n) \approx \boxed{P(w_1|<s>\ <s>)P(w_2|<s>\ w_1)}...P(w_n|w_{n-2}\ w_{n-1})$$

- N-gram model: add N-1 start tokens <s>

End of sentence token <s> - motivation

$$P(y|x) = \frac{C(x\ y)}{\boxed{\sum_w C(x\ w)}} = \frac{C(x\ y)}{\boxed{C(x)}}$$

Corpus:

<s> Lyn $\boxed{\text{drinks}}$ $\boxed{\text{chocolate}}$

<s> John $\boxed{\text{drinks}}$

$$\sum_w C(drinks\ w) = 1$$

$$C(drinks) = 2$$

End of sentence token <s> - motivation (cont.)

KAUST ACADEMY أكاديمية كاوست

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 2:

<s> yes yes

<s> yes no

<s> no no

<s> no yes

$$P(<s> \text{ yes yes}) =$$

$$P(\text{yes} \mid <s>) \times P(\text{yes} \mid \text{yes}) =$$

$$\frac{C(<s> \text{ yes})}{\sum_w C(<s> w)} \times \frac{C(\text{yes yes})}{\sum_w C(\text{yes } w)} =$$

$$\frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$$

End of sentence token <s> - motivation (cont.)

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 2:

<s> yes yes

<s> yes no

<s> no no

<s> no yes

$$P(<s> \text{ yes yes}) = \frac{1}{3}$$

$$P(<s> \text{ yes no}) = \frac{1}{3}$$

$$P(<s> \text{ no no}) = \frac{1}{3}$$

$$P(<s> \text{ no yes}) = 0$$

$$\sum_{2 \text{ word}} P(\cdots) = 1$$

End of sentence token <s> - motivation (cont.)

Corpus

<s> yes no

<s> yes yes

<s> no no

Sentences of length 3:

<s> yes yes yes

<s> yes yes no

...

<s> no no no

$$P(< s > \ \text{yes yes yes}) = \dots$$

$$P(< s > \ \text{yes yes no}) = \dots$$

$$\dots \qquad = \dots$$

$$P(< s > \ \text{no no no}) = \dots$$

$$\sum_{3 \text{ word}} P(\cdots) = 1$$

End of sentence token <s> - motivation (cont.)

Corpus

<s> yes no

<s> yes yes

<s> no no

$$\sum_{2\ \text{word}} P(\cdots) + \sum_{3\ \text{word}} P(\cdots) + \ldots = 1$$

End of sentence token <s> - solution

- Bigram

<s> the teacher drinks tea => <s> the teacher drinks tea </s>

$$P(the|<s>)P(teacher|the)P(drinks|teacher)P(tea|drinks)\boxed{P(</s>|tea)}$$

Corpus:

<s> Lyn **drinks chocolate** </s>

<s> John **drinks </s>**

$$\sum_w C(drinks\ w) = 2$$

$$C(drinks) = 2$$

End of sentence token <s> for N-grams

- N-gram => just one </s>

  E.g. Trigram:

  the teacher drinks tea => <s> <s> the teacher drinks tea </s>

Example - bigram

Corpus

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

$$P(sentence) = \boxed{\frac{2}{3}} * \boxed{\frac{1}{2}} * \boxed{\frac{1}{2}} * \boxed{\frac{2}{2}} = \frac{1}{6}$$

$$P(John|<s>) = \frac{1}{3}$$

$$P(</s>|tea) = \frac{1}{1}$$

$$P(chocolate|eats) = \frac{1}{2}$$

$$P(Lyn|<s>) =? = \frac{2}{3}$$

Quiz

Objective: Apply sequence probability approximation with bigrams after adding start and end word.

Question:
Given these conditional probabilities

P(Mary)=0.1;	P(likes)=0.2;	P(cats)=0.3
P(Mary|<s>)=0.2;	P(</s>|cats)=0.6	
P(likes|Mary) =0.3;	P(cats|likes)=0.1	

Approximate the probability of the following sentence with bigrams: "<s> Mary likes cats </s>"
Type: Multiple Choice, single answer

Options and solution:

1. P(<s> Mary likes cats </s>) = 0
2. P(<s> Mary likes cats </s>) =0.0036
3. P(<s> Mary likes cats </s>) = 0.003
4. P(<s> Mary likes cats </s>) = 1

N-gram Models: **Out of vocabulary words**

Out of Vocabulary (OOV) Words

Problem: Many words in a language are not present in the training corpus, leading to OOV issues.

Out of Vocabulary (OOV) Words

Problem: Many words in a language are not present in the training corpus, leading to OOV issues.

Example: The word "quokka" might not be in the training data.

Out of Vocabulary (OOV) Words

Problem: Many words in a language are not present in the training corpus, leading to OOV issues.

Example: The word "quokka" might not be in the training data.

Impact: OOV words can lead to poor model performance and inaccurate predictions.

Out of Vocabulary (OOV) Words

Problem: Many words in a language are not present in the training corpus, leading to OOV issues.

Example: The word "quokka" might not be in the training data.

Impact: OOV words can lead to poor model performance and inaccurate predictions.

Closed vs. Open Vocabularies:

- **Closed vocabulary:** Only words seen during training are recognized.
- **Open vocabulary:** Model can handle unseen words.

Out of Vocabulary (OOV) Words

Problem: Many words in a language are not present in the training corpus, leading to OOV issues.

Example: The word "quokka" might not be in the training data.

Impact: OOV words can lead to poor model performance and inaccurate predictions.

Closed vs. Open Vocabularies:

- **Closed vocabulary:** Only words seen during training are recognized.
- **Open vocabulary:** Model can handle unseen words.

Solution: Use a special tag `<UNK>` in the corpus and input to represent unknown words.

Using <UNK> Tokens

Using `<UNK>`: Replace rare or unseen words with the special token `<UNK>`.

Using <UNK> Tokens

Using `<UNK>`: Replace rare or unseen words with the special token `<UNK>`.

Why? Helps the model generalize to words it has not seen during training.

Using <UNK> Tokens

Using `<UNK>`: Replace rare or unseen words with the special token `<UNK>`.

Why? Helps the model generalize to words it has not seen during training.

Example:

- Original: `I love ChatGPT`
- With OOV handling: `I love <UNK>`

How to Use <UNK> in the Corpus

1. **Create vocabulary V:** Build a list of all words to be recognized (e.g., most frequent words).
2. **Replace OOV words:** For any word in the corpus not in V, replace it with `<UNK>`.
3. **Estimate probabilities:** Treat `<UNK>` as a regular word when computing word probabilities.

This approach allows the model to handle unseen words gracefully during inference.

Example

Corpus

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

Corpus

<s> Lyn drinks chocolate </s>

<s> <UNK> drinks <UNK> </s>

<s> Lyn <UNK> chocolate </s>

Min frequency f=2

Vocabulary

Lyn, drinks, chocolate

Input query

<s> Adam drinks chocolate</s>

<s><UNK> drinks chocolate</s>

How to Create Vocabulary V

Criteria for Vocabulary Selection:

- **Minimum word frequency f:** Only include words that appear at least f times in the corpus.
- **Maximum vocabulary size $|V|$:** Limit V to the top $|V|$ most frequent words.

Use `<UNK>` Sparingly: Choose f and $|V|$ to minimize the number of words replaced by `<UNK>`, while keeping the vocabulary manageable. **Perplexity:** Only

compare language models that use the same vocabulary V to ensure fair evaluation.

N-gram Models: **Limitations**

N-gram Model Limitations

Data Sparsity

Limited Context (only N-1 words)

Explodes with Vocabulary Size

Doesn't Capture Semantics/Syntax

N-gram Models: **Summary**

Beyond N-grams

- Neural Language Models (e.g., Word2Vec, LSTMs)
- Transformer-based Models (BERT, GPT)
- Subword Tokenization (Byte-Pair Encoding)
- Pretrained Language Models
- Contextual Representations

Summary

- NLP enables machines to understand human language
- N-grams model word sequences simply and effectively
- Sequence notation and probability estimation are essential
- Start/end/UNK tokens improve modeling and robustness
- N-gram models are limited $\rightarrow$ neural models offer solutions

NLP: **References**

References

- Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing (3rd Ed Draft)* – https://web.stanford.edu/~jurafsky/slp3/
- CMU NLP Course Slides – https://www.cs.cmu.edu/~tom/mlbook/NLP.html
- Stanford CS224N (2023) – https://web.stanford.edu/class/cs224n/
- Manning, C. D., et al. (2008). *Introduction to Information Retrieval*
- Bengio et al. (2003) – A Neural Probabilistic Language Model, JMLR
- Mikolov et al. (2013) – Efficient Estimation of Word Representations in Vector Space
- Vaswani et al. (2017) – Attention is All You Need
- Younes Mourri & Lukasz Kaiser, *Natural Language Processing Specialization*, DeepLearning.AI – https://www.deeplearning.ai/courses/natural-language-processing-specialization/

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.