MTH 227

Applied Natural Language Processing

Batuhan Bardak

Lecture 1: Course outline and basic concepts of NLP

Date: 03.10.2025

About Me

Dr. Batuhan Bardak

- 2014, 2016, 2022 B.Sc., M.Sc., Ph.D., TOBB ETU Computer Science
- 2014-2017 TOBB ETU, Teaching & Research Assistant
- 2018 M.Sc, METU, Engineering Management
- 2017- .. STM, Senior Lead Data Scientist

Plan for today

- Course outline and materials
- Basic concepts and terminology of Natural Language Processing

Logistics

- **Instructor**: Batuhan BARDAK
  - batuhanbardak@gmail.com
  - bbatuhan@metu.edu.tr
- **Lectures**:
- **TA:**
  -
- **ODTU Class:** https://odtuclass2025f.metu.edu.tr/user/index.php?id=3986
- **Github**
  - https://github.com/bbardakk/METU-MTH-227-Applied-NLP

Prerequisites

- Basic algorithms and data structures
- Good programming skills (especially in Python)
- Good machine learning fundamental

Course outline (Tentative)

- **Week 1:** Introduction to NLP and Text Preprocessing
- **Week 2:** Text to Vector & Embeddings
- **Week 3:** NLP Applications
- **Week 4**: Neural Networks for NLP
- **Week 5**: CNN & RNN for NLP
- **Week 6:** The Attention Mechanism and Transformer Architecture
- **Week 7:** BERT and Variants
- **Week 8:** Introduction to Large Language Models (LLMs)
- **Week 9:** LLMs II
- **Week 10:** Prompt Engineering and In-Context Learning
- **Week 11**: Fine-Tuning LLMs
- **Week 12**: LLM Applications and Vector Databases
- **Week 13**: Retrieval Augmented Generation (RAG)
- **Week 14**: Project Presentations

Grading

- Final: %30 (closed book, no notes, no cheat sheet)
- Homeworks: %30 (with Python)
- Course Project: %40 (done in groups of 3)
- Attendance: %5 (To pass the course, attendance to 50% of the classes is mandatory.)

Course Project

- Propose or Select a project topic (real-world use case) (Including vector databases, LLMs etc.)

- Projects should be done in groups (2-3 people).

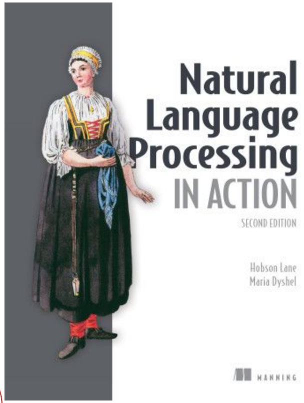
- Grading
  - Proposal (% 5)
  - Literature Review (% 5)
  - Novelty (%10)
  - Progress Report (%10)
  - Deployment (%20)
  - Project Presentation (%20)
  - Final Report (%30)

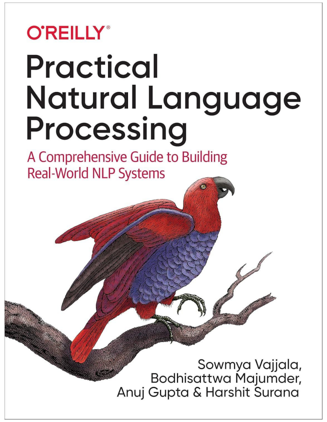
Academic Integrity

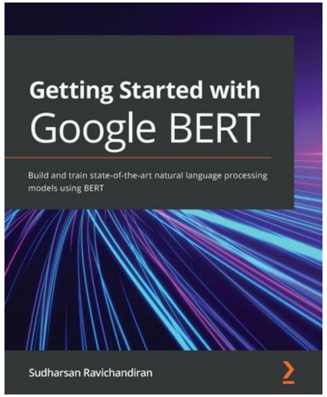
- **Zero tolerance policy:** All occurrences will be reported
- **AI tool policy:** You can use AI-based tools, but you need to provide references. You also need to mention how you used this tool (prompts, etc.) and its contribution to you.

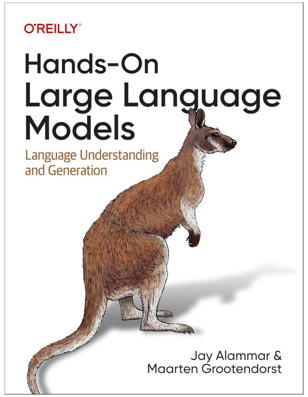
Reference Books

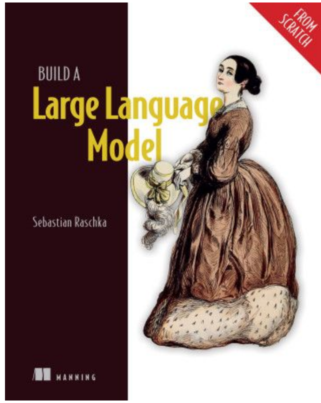
- Natural Language Processing in Action
- Practical Natural Language Processing
- Getting started with Google BERT
- Hands-On Large Language Models
- Large Language Model from scratch


Natural Language Processing IN ACTION
SECOND EDITION
Hobson Lane
Maria Dyshel
MANNING


O'REILLY
Practical Natural Language Processing
A Comprehensive Guide to Building Real-World NLP Systems
Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta & Harshit Surana


Getting Started with Google BERT
Build and train state-of-the-art natural language processing models using BERT
Sudharsan Ravichandiran


O'REILLY
Hands-On Large Language Models
Language Understanding and Generation
Jay Alammar & Maarten Grootendorst


FROM SCRATCH
BUILD A Large Language Model
Sebastian Raschka
MANNING

Today's Journay

Part 1: The World of NLP

- What is Natural Language Processing?
- A Brief History: From Rules to Neural Networks
- Key Milestones & Game Changers in NLP History
- The Fundamental Challenges of Language

Part 2: Recap Machine Learning

Part 3: Preparing Text for Machines

- The Text Preprocessing Pipeline & Core Steps
- Choosing Your Preprocessing Strategy: When and Why?
- Code Examples

What is Natural Language Processing (NLP)?

- **Natural Language Processing (NLP)** is a subfield of Artificial Intelligence (AI), Computer Science, and Linguistics concerned with the interactions between computers and human (natural) languages.

- **The Ultimate Goal:** To enable computers to understand, interpret, generate, and respond to human language in a way that is both meaningful and useful.

https://www.ibm.com/think/topics/natural-language-processing#:~:text=Natural%20language%20processing%20,and%20communicate%20with%20human%20language

Brief History of NLP

The Evolution of NLP: Three Major Eras

Symbolic NLP (1950s - 1990s)

- **Approach:** Relied on hand-crafted rules, grammars, and lexicons created by linguists.
- **Example:** "If you see a word ending in '-ing', it is likely a verb."
- **Limitation:** Brittle, not robust to exceptions, and required immense human effort.

Brief History of NLP

Statistical NLP (1990s - 2010s)

- **Approach:** Used machine learning models (like Naive Bayes, Support Vector Machines) trained on large text corpora to make probabilistic decisions.
- **Example:** Based on a million sentences, the word "bank" is 95% likely to be a noun.
- **Advantage:** More robust and scalable than rule-based systems.

Brief History of NLP

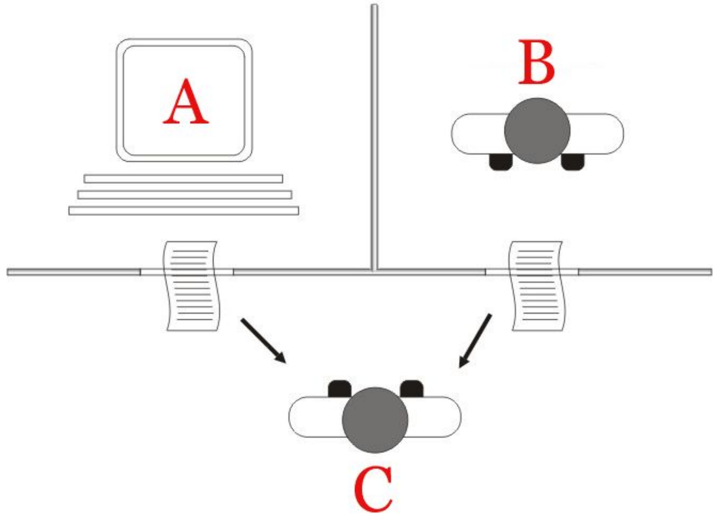
Neural NLP (2010s - Present)

- **Approach:** Utilizes deep learning models (RNNs, LSTMs, Transformers) to learn feature representations directly from data.
- **Advantage:** Achieved state-of-the-art performance by capturing complex patterns and context without manual feature engineering. This is the focus of our course.

NLP Milestones: The Foundational Years

1950: The Turing Test

- Alan Turing's proposal becomes a foundational concept for measuring machine intelligence in language.



NLP Milestones: The Foundational Years

1954: Georgetown - IBM Experiment

- One of the first public demonstrations of Machine Translation (MT), translating sentences into English. It was heavily rule-based and limited, but sparked immense interest.

Fig. 2: Hurd, Dostert and Watson at the demonstration

NLP Milestones: The Foundational Years

1966: ELIZA

- An early chatbot created by Joseph Weizenbaum that used pattern matching to simulate a conversation, demonstrating the illusion of understanding.
    a. https://web.njit.edu/~ronkowit/eliza.html

NLP Milestones: The Foundational Years

1990s: Rise of Statistical Methods

- The field shifts from hand-crafted rules to learning from data. **Hidden Markov Models** (HMMs) become standard for **Part-of-Speech** (POS) tagging. **Statistical Machine Translation** (SMT) begins to dominate.

NLP Milestones: The Foundational Years

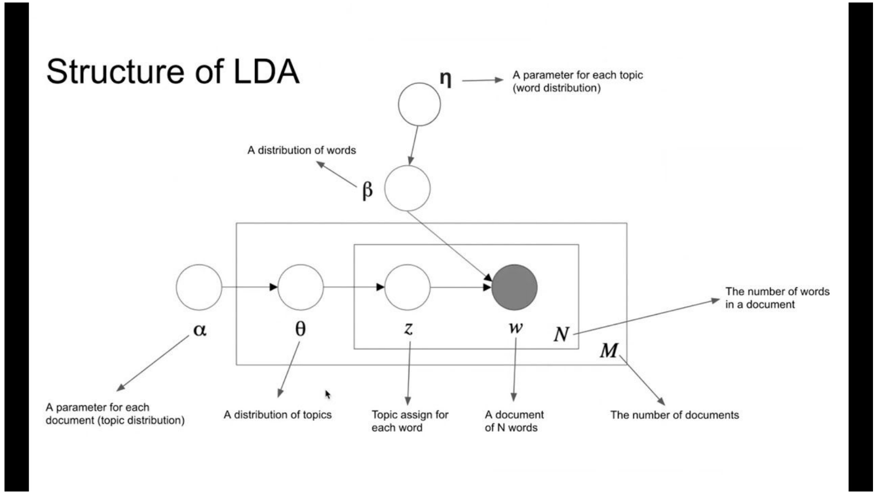
2001: Latent Dirichlet Allocation (LDA)

- David Blei, Andrew Ng, and Michael I. Jordan’s paper introduces a generative statistical model for discovering abstract “topics” in a collection of documents.

NLP Milestones: The Foundational Years

2001: Latent Dirichlet Allocation (LDA)

- David Blei, Andrew Ng, and Michael I. Jordan's paper introduces a generative statistical model for discovering abstract "topics" in a collection of documents.

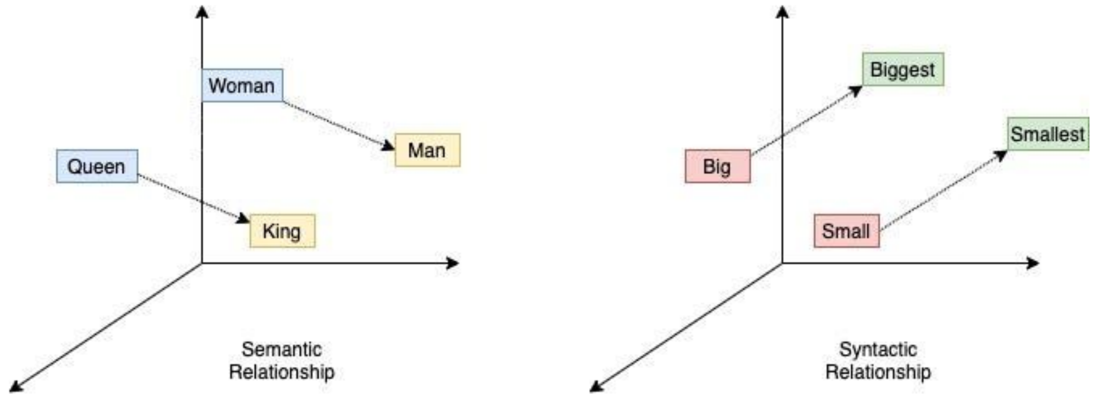


Structure of LDA
η
A parameter for each topic (word distribution)
A distribution of words
β
α
θ
z
w
N
M
The number of words in a document
A parameter for each document (topic distribution)
A distribution of topics
Topic assign for each word
A document of N words
The number of documents

NLP Milestones: The Foundational Years

2013: Word2Vec

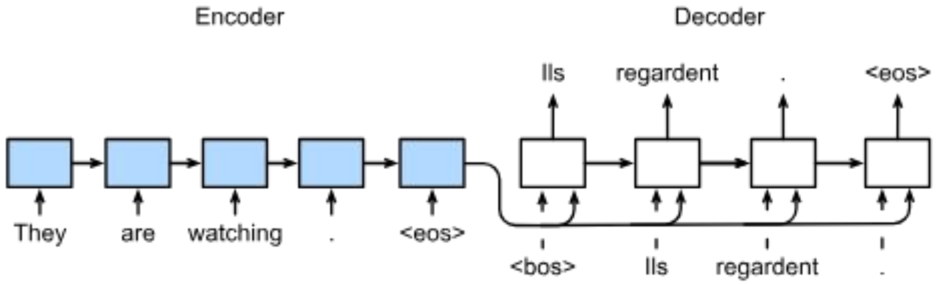
- Tomas Mikolov et al. at Google release a model that learns dense vector representations of words (**word embeddings**). This was a revolutionary shift, allowing models to capture semantic relationships (e.g. king - man + woman = queen)



NLP Milestones: The Deep Learning Revolution

2014: Sequence-to-Sequence (Seq2Seq) with RNNs/LSTMs

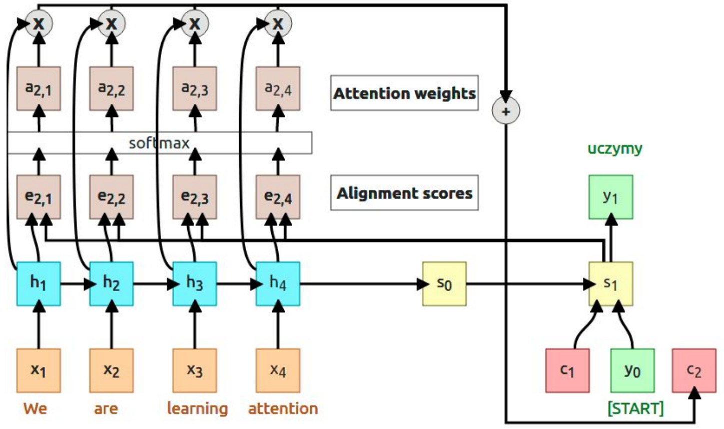
- Sutskever et al. show that Recurrent Neural Networks (RNNs), particularly LSTMs, can effectively map an input sequence to an output sequence, achieving new state-of-the-art results in Machine Translation.



NLP Milestones: The Deep Learning Revolution

2014: The Attention Mechanism

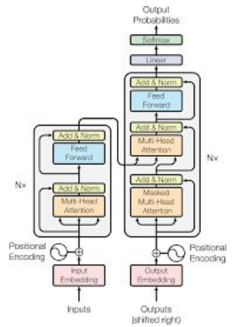
- Bahdanau et al. introduce “attention” to solve the bottleneck problem in Seq2Seq model. Instead of relying on a single context vector, the model can “pay attention” to relevant parts of the input sequence while generating output. This concept is fundamental to modern NLP.



NLP Milestones: The Deep Learning Revolution

2017: The Transformer: “Attention Is All You Need”

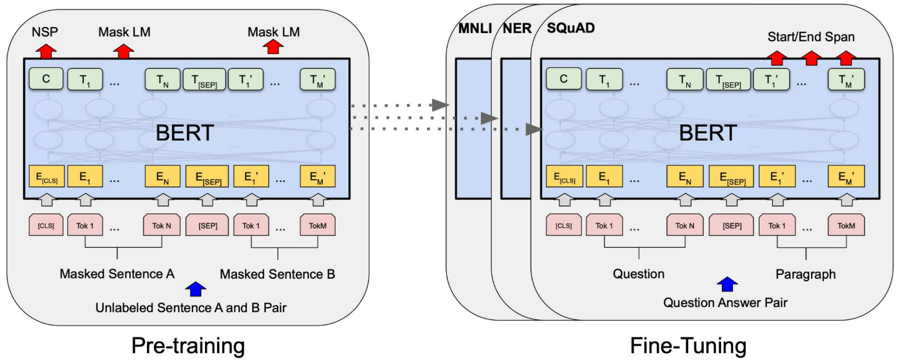
- A landmark paper from Google (Vaswani et al.) introduces the **Transformer architecture**. It completely dispenses with recurrence and relies solely on self-attention mechanisms.
- This architecture is more parallelizable, efficient, and powerful, becoming the bedrock for nearly all subsequent subsequent SoTA models.



NLP Milestones: The Age of Transformers & LLMs

2018: BERT: Pre-training of Deep Bidirectional Transformers

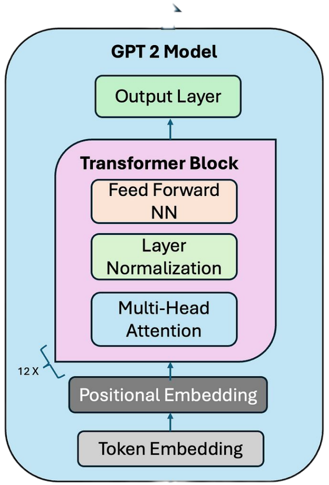
- Google's BERT model revolutionizes the field by introducing transfer learning on a massive scale. It's pre-trained on a huge corpus of text and can be quickly fine-tuned for various downstream tasks, achieving SoTA on 11 NLP Benchmarks.



NLP Milestones: The Age of Transformers & LLMs

2019: GPT-2

- OpenAI releases GPT-2, a large scale (1.5 billion parameters) decoder-only Transformer model. Its impressive ability to generate coherent, human-like text brings the power of generative models to the public consciousness.



NLP Milestones: The Age of Transformers & LLMs

2020: GPT-3

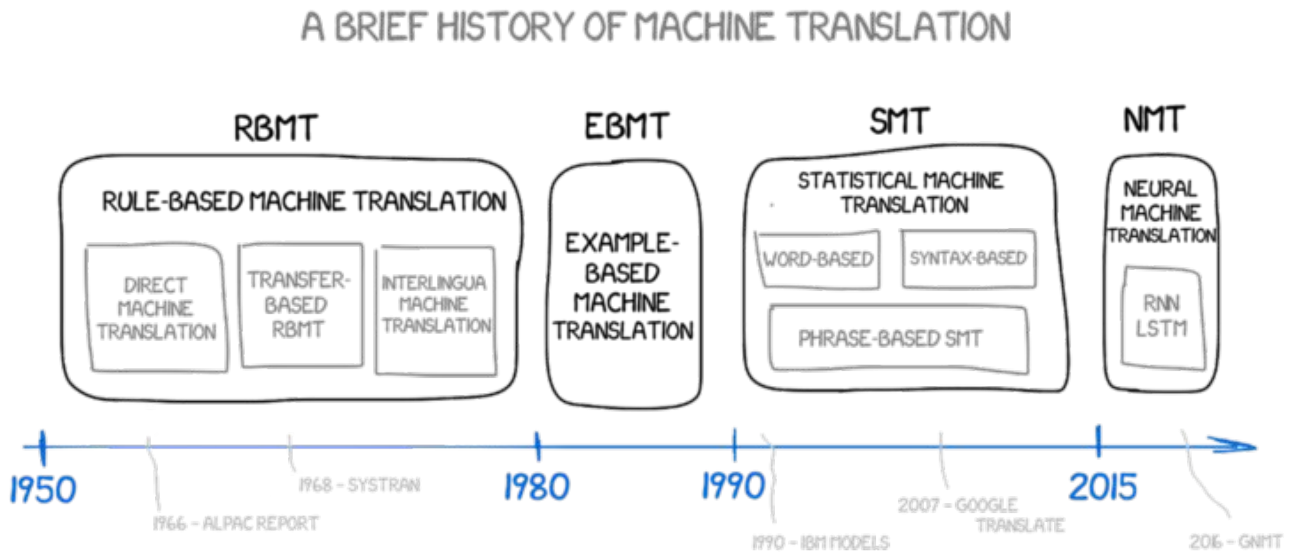
- With 175 billion parameters, GPT-3 demonstrates remarkable **in-context learning** (few shot learning) capabilities, performing tasks with high accuracy from just a few examples in the prompt, without any fine-tunning.

NLP Milestones: The Age of Transformers & LLMs

2022: The Rise of Instruction Tuning (e.g., ChatGPT)

- The focus shifts to aligning LLMs with human intent. Models are fine-tuned on human written instructions and further refined using Reinforcement Learning from Human Feedback (RLHF), making them vastly more helpful, safe, and conversational.

NLP Milestones: The Foundational Years


A BRIEF HISTORY OF MACHINE TRANSLATION
RBMT
RULE-BASED MACHINE TRANSLATION
DIRECT MACHINE TRANSLATION
TRANSFER-BASED RBMT
INTERLINGUA MACHINE TRANSLATION
EBMT
EXAMPLE-BASED MACHINE TRANSLATION
SMT
STATISTICAL MACHINE TRANSLATION
WORD-BASED
SYNTAX-BASED
PHRASE-BASED SMT
NMT
NEURAL MACHINE TRANSLATION
RNN
LSTM
1950
1966 - ALPAC REPORT
1968 - SYSTRAN
1980
1990
1990 - IBM MODELS
2007 - GOOGLE TRANSLATE
2015
2016 - GNMT

Why is NLP Important? Core Applications

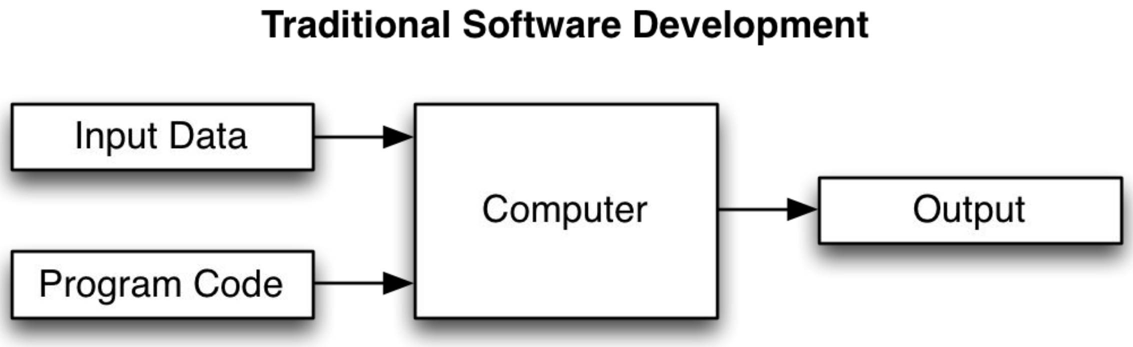
NLP is Everywhere

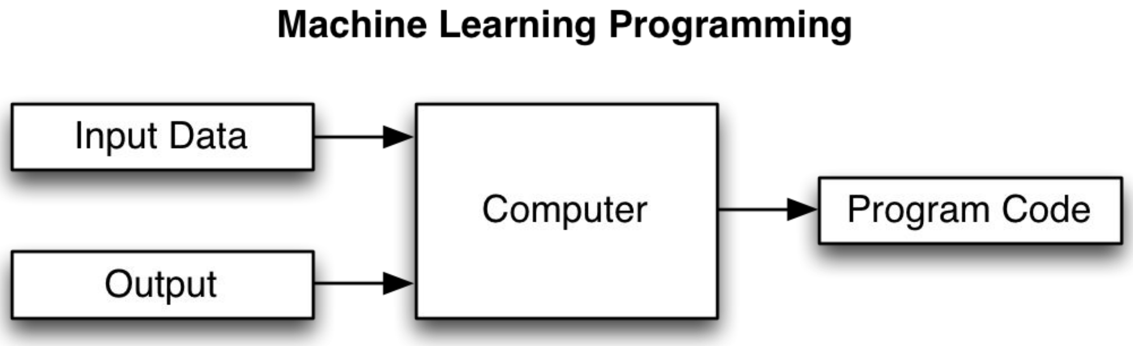
- **Search Engines:** Google, Bing (Understanding your query, ranking results).
- **Virtual Assistants:** Siri, Alexa, Google Assistant (Speech recognition, intent understanding).
- **Machine Translation:** Google Translate (Translating text between languages).
- **Sentiment Analysis:** Analyzing social media posts or reviews to determine if they are positive, negative, or neutral.
- **Chatbots & Conversational AI:** Customer service bots, automated helpdesks.
- **Text Summarization:** Condensing long articles into short summaries.
- **Spam Detection:** Filtering unwanted emails.

Recap of Machine Learning

Figure:

Traditional Programming vs Machine Learning



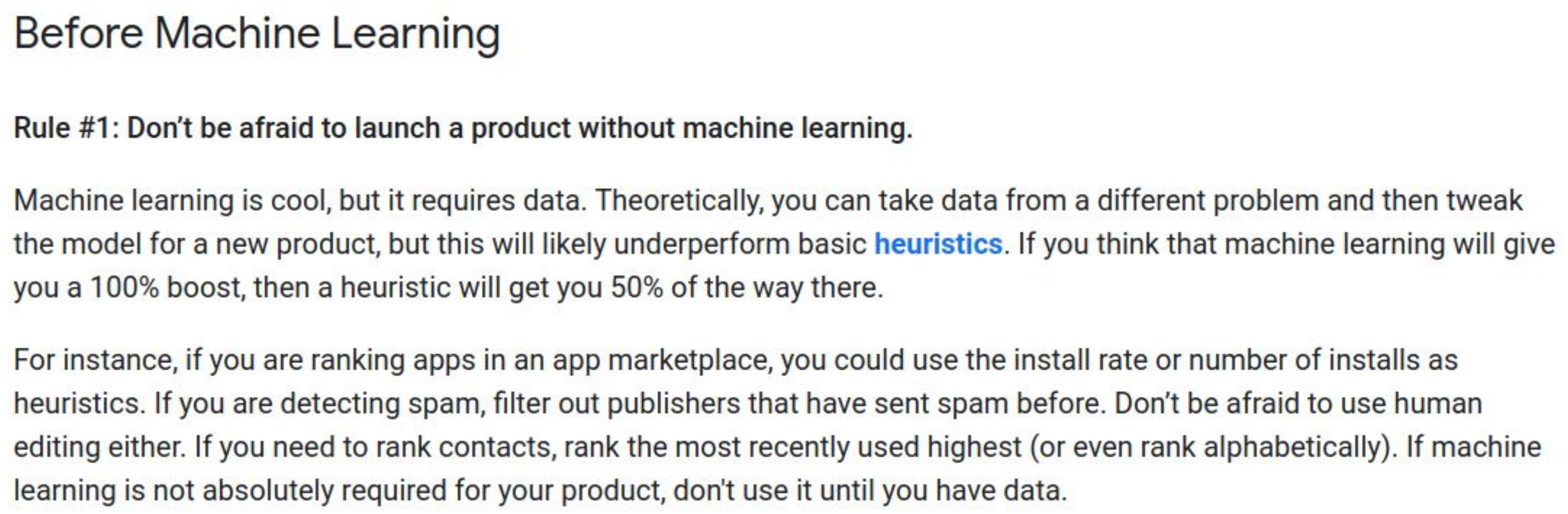


Machine Learning in a Nutshell

- Every machine learning algorithm consists of the following basic steps:
  - Data collection
  - Representation
  - Modeling
  - Evaluation
  - Optimization

Rules of Machine Learning

- https://developers.google.com/machine-learning/guides/rules-of-ml



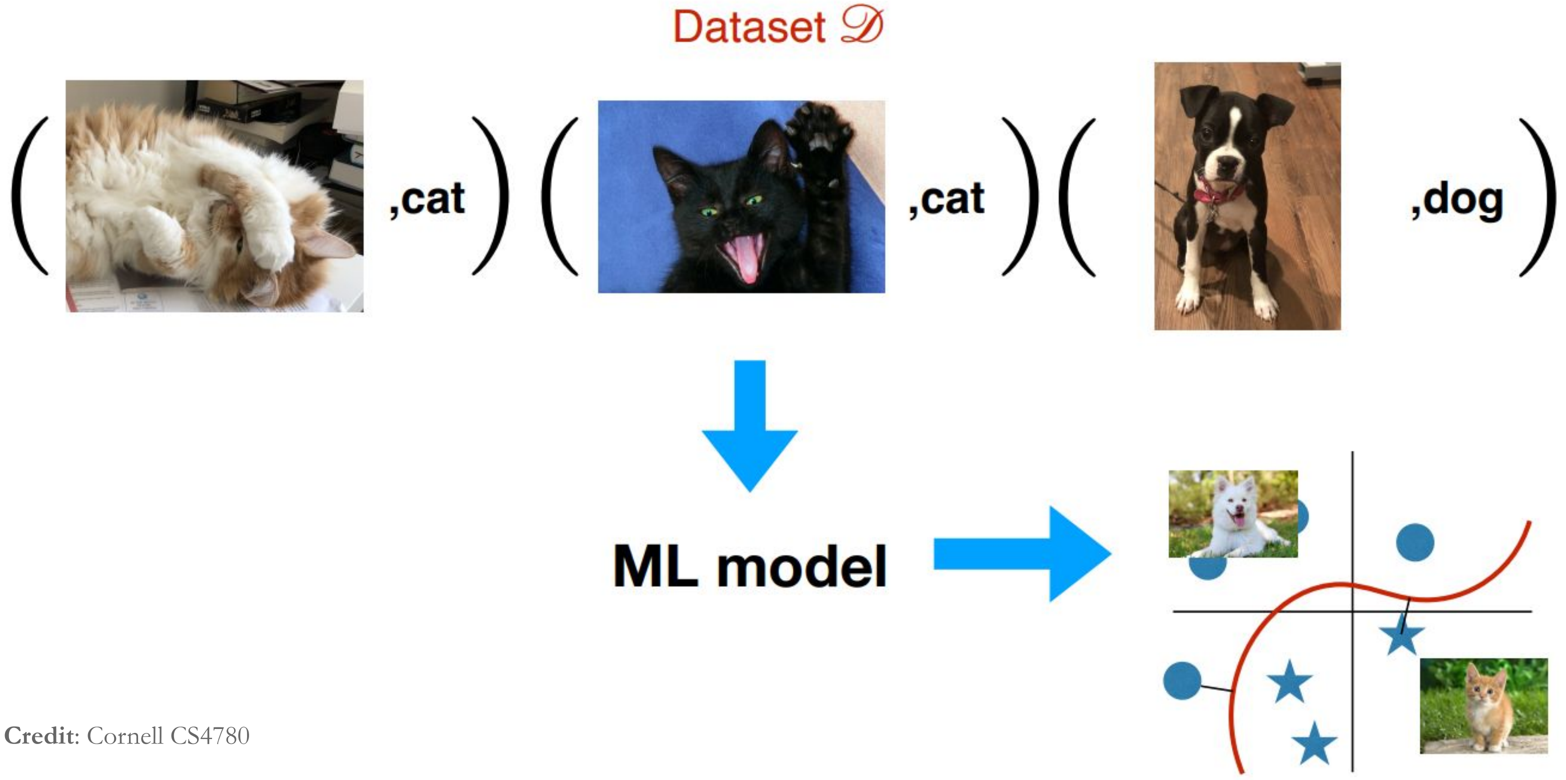
Before Machine Learning

Rule #1: Don't be afraid to launch a product without machine learning.

Machine learning is cool, but it requires data. Theoretically, you can take data from a different problem and then tweak the model for a new product, but this will likely underperform basic **heuristics**. If you think that machine learning will give you a 100% boost, then a heuristic will get you 50% of the way there.

For instance, if you are ranking apps in an app marketplace, you could use the install rate or number of installs as heuristics. If you are detecting spam, filter out publishers that have sent spam before. Don't be afraid to use human editing either. If you need to rank contacts, rank the most recently used highest (or even rank alphabetically). If machine learning is not absolutely required for your product, don't use it until you have data.

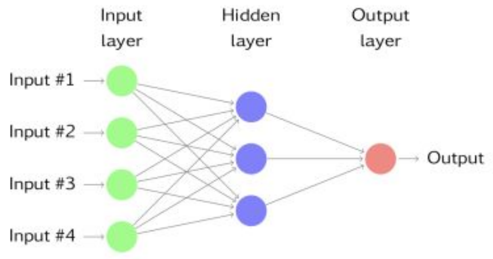
Classification



Mathematical formulation of the pipeline

Dataset:

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathcal{C}(\text{ e.g.,}\mathcal{C} = \{-1,1\}), (x_i, y_i) \sim \mathcal{P}$$



Hypothesis:

$$h : \mathbb{R}^d \mapsto \mathcal{C}$$

i.e., a neural network-based classifier that maps image to label of cat or dog

Hypothesis class

$$\mathcal{H} = \{h\}$$

i.e., a large family of NNs with different parameters

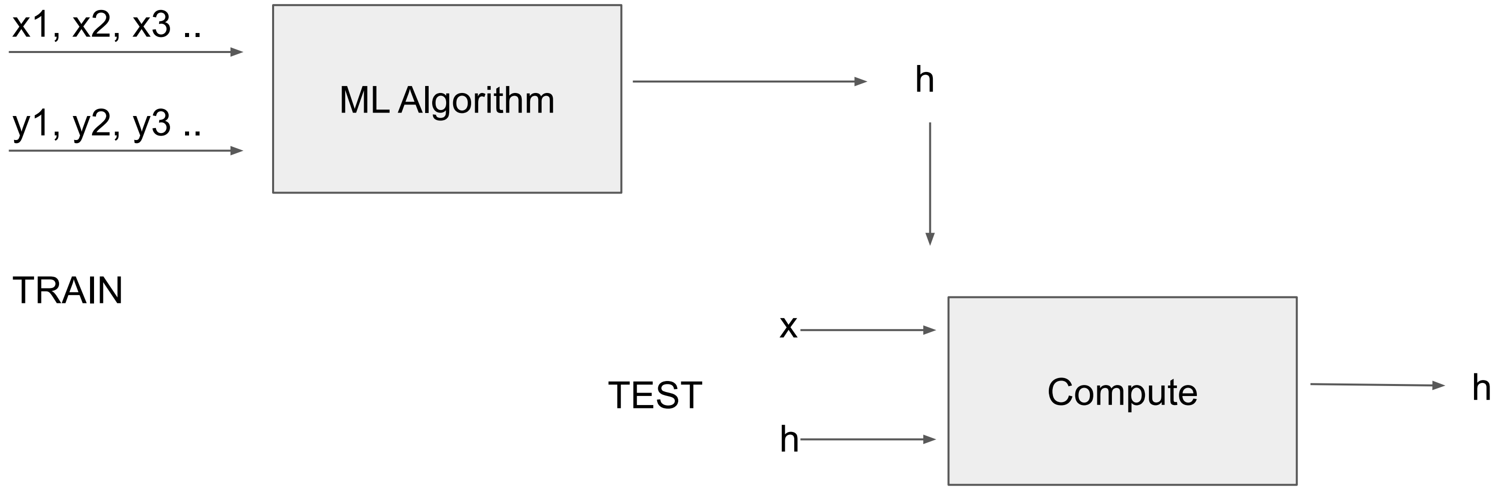
Type of Y

- Binary Classification: Y = {0, 1}, Y = {-1, 1}
- Multi-Class Classification: Y = {1, 2, 3 .. K}
- Regression: Y = Real Numbers

Type of X

- Patient Data
- Text Documents
- Images

Training

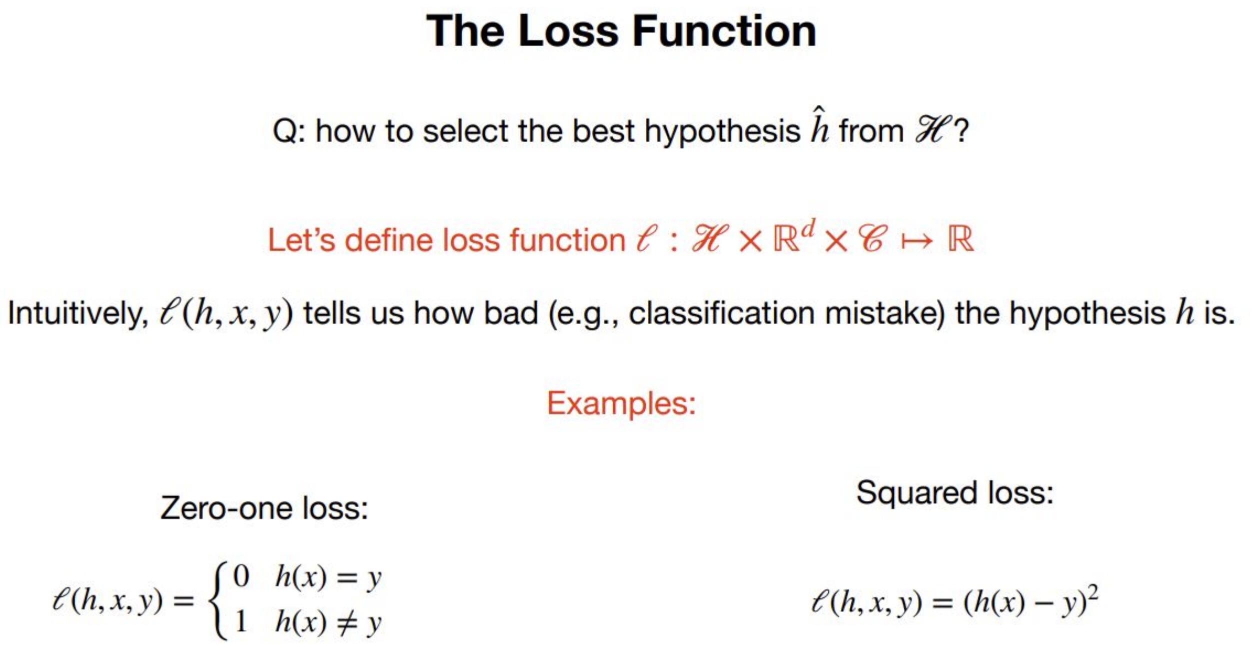


Learning

- Algorithm 1:
  - Pick h € H randomly
  - What is the problem?

- Algorithm 2:
  - Try every single h in H to find best one
  - What is the problem?

The Loss Function



The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

Let's define loss function $\ell : \mathscr{H} \times \mathbb{R}^d \times \mathscr{C} \mapsto \mathbb{R}$

Intuitively, $\ell(h, x, y)$ tells us how bad (e.g., classification mistake) the hypothesis h is.

Examples:

Zero-one loss:

$$\ell(h,x,y) = \begin{cases} 0 & h(x) = y \\ 1 & h(x) \neq y \end{cases}$$

Squared loss:

$$\ell(h,x,y) = (h(x) - y)^2$$

Slide Credit: Cornell CS4780

The Loss Function

Learning/Training

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

With loss ℓ being defined, we can perform training/learning:

$$\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h, x_i, y_i)$$

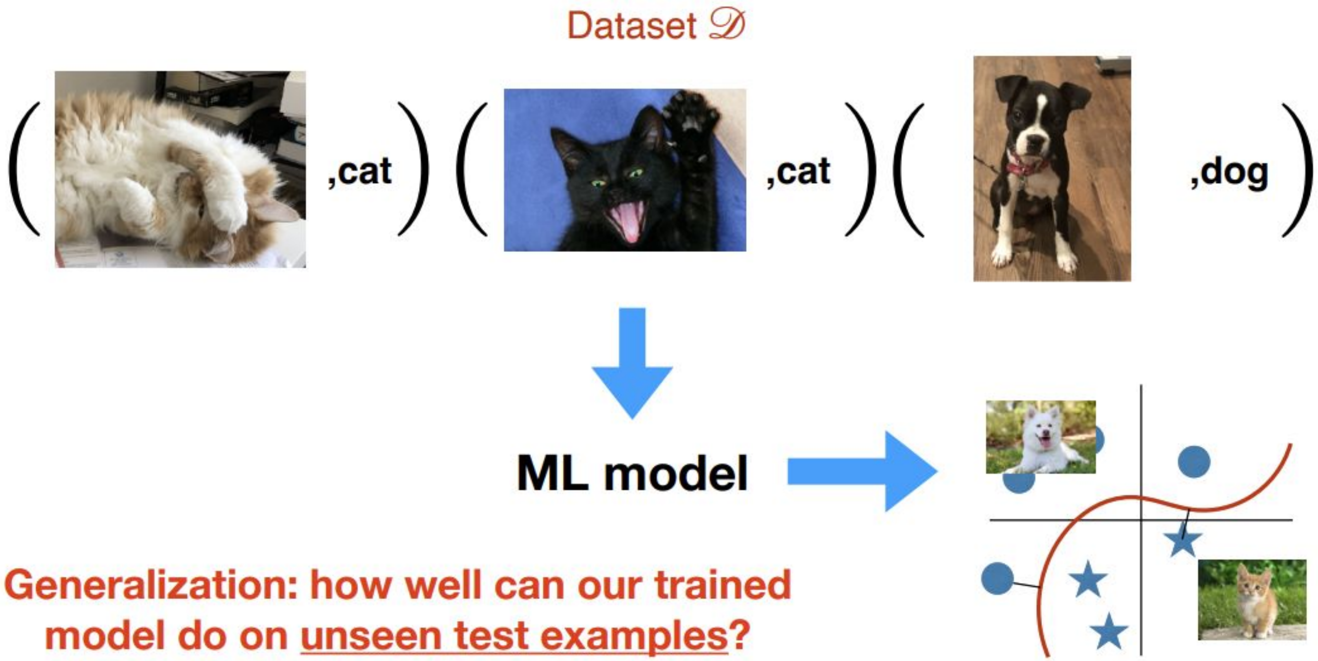
Learning

- Algorithm 3:

$$h(x) = \begin{cases} y_i & x = x_i \text{ for } (x_i, y_i) \in D \\ y_1 & \text{otherwise} \end{cases}$$

  - What is the problem?

Generalization



Generalization: how well can our trained model do on unseen test examples?

Slide Credit: Cornell CS4780

Formalize with using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Training data $\mathscr{D}$ is i.i.d sampled from a distribution $\mathscr{P}$, i.e., $x_i, y_i \sim \mathscr{P}, \forall i \in [n]$
(i.e., all pairs are sampled from $\mathscr{P}$, and (x_i, y_i) is independent of others)

We further assume test data is also from $\mathscr{P}$, i.e., $(x, y) \sim \mathscr{P}$

Generalization error: $\mathbb{E}_{x,y\sim\mathscr{P}}\left[\ell(\hat{h}, x, y)\right]$

e.g., expected classification error of $\hat{h}$

Overfitting

Overfitting: we have a small training error but large generalization error

Example

Hypothesis $\tilde{h}$ that memorizes the whole training set

$$\tilde{h}(x) = \begin{cases} y_i & \exists (x_i, y_i) \in \mathscr{D} \text{ w/ } x_i = x \\ 0 & \text{else} \end{cases}$$

Overfitting

Overfitting: we have a small training error but large generalization error

Example

Hypothesis $\tilde{h}$ that memorizes the whole training set

$$\tilde{h}(x) = \begin{cases} y_i & \exists (x_i, y_i) \in \mathscr{D} \text{ w/ } x_i = x \\ 0 & \text{else} \end{cases}$$

Training error = 0, but could do terribly on test examples

Overfitting

Overfitting: we have a small training error but large generalization error

Example

Hypothesis $\tilde{h}$ that memorizes the whole training set

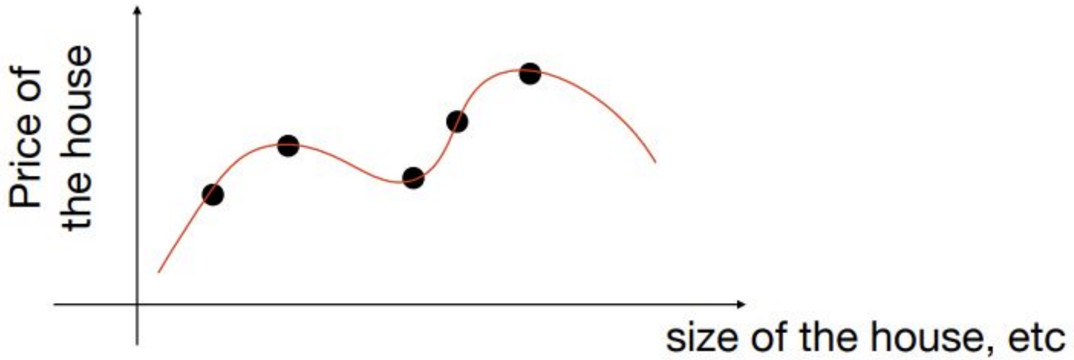
$$\tilde{h}(x) = \begin{cases} y_i & \exists (x_i, y_i) \in \mathscr{D} \text{ w/ } x_i = x \\ 0 & \text{else} \end{cases}$$

Training error = 0, but could do terribly on test examples

Overfitting

Overfitting: we have a small training error but large generalization/test error

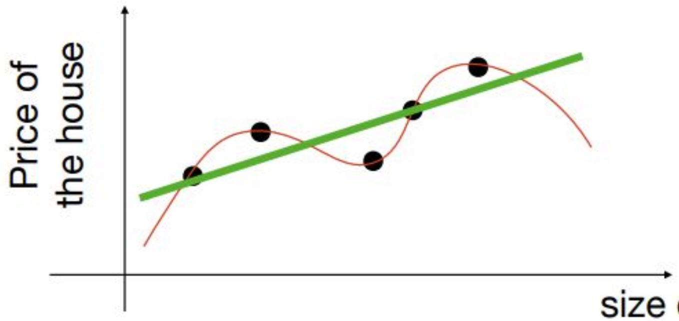
Example



Overfitting

Overfitting: we have a small training error but large generalization/test error

Example



Training error = 0 (e.g., we probably overfit to noises), but could do terribly on test examples

Training, validation, and testing

Given a training dataset $\mathscr{D}$, we can split it into three sets:

$\mathscr{D}_{TR}$: training set

$\mathscr{D}_{VA}$: validation set

$\mathscr{D}_{TE}$: test set

Before training/learning, we often **randomly** split it with size proportional to 80% / 10% / 10%

Selecting models using validation set

We can use validation set to select models, i.e., select hypothesis class, tune parameters, etc

Small avg error on $\mathscr{D}_{TR}$ but larger avg error on $\mathscr{D}_{VA}$ indicates overfitting

$$\Downarrow$$

Revise model on $\mathscr{D}_{TR}$ (e.g., add regularization, change neural network structures, etc)

Do not use test set to train/select models

We should not touch test set during training!

This makes sure that the test set $\mathscr{D}_{TE}$ is independent of our model $\hat{h}$

Such independence implies that:

$$\frac{1}{|\mathscr{D}_{TF}|} \sum_{x,y \in \mathscr{D}_{TE}} \ell(\hat{h}, x, y) \approx \mathbb{E}_{x,y \sim \mathscr{P}}[\ell(\hat{h}, x, y)]$$

(Due to law of large numbers)

Slide Credit: Cornell CS4780

Do not use test set to train/select models

What if our original dataset is quite small, e.g., $n = 100$
(very possible in medical applications!)

K-fold cross validation

Split the data into K folds (e.g., K = 10 or 20)

```
For $i = 1 \rightarrow K$:
    $\mathscr{D}_{TR}$: all others folds except the i'th fold
    Train model $\hat{h}$ on $D_{TR}$
    Validate on the i'th fold (i.e., $\mathscr{D}_{VR}$ = i'th fold)
Average K validation errors
```

When K = n, this is leave-one-out cross validation

Summary

1. Given a task and a dataset
$\mathscr{D} = \{x_i, y_i\}, x_i, y_i \sim \mathscr{P}$

2. Design hypothesis class $\mathscr{H}$ and loss function ℓ (encodes inductive bias)

3. Train: $\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{(x,y \in \mathscr{D})} \ell(h, x, y)$

Often repeated many times using $\mathscr{D}_{VA}$ / cross validation

Output: $\hat{h}$ that has small generalization error
$\mathbb{E}_{x,y \sim \mathscr{P}}[\ell(\hat{h}, x, y)]$

The Fundamental Challenges of Language

Why is Language So Hard for Computers?

- **Ambiguity:** The same word or phrase can have multiple meanings.
  - *Lexical Ambiguity:* "I went to the **bank.**" (River bank or financial institution?)
  - *Syntactic Ambiguity:* “I saw a man on a hill with a telescope.”
    1. Did I use telescope?
    2. Did the man have it?
    3. Was the man on the hill or did I see from a hill?
- **Synonymy:** Different words can mean the same thing.
  - "big" vs. "large" OR "buy" vs. "purchase"
- **Context is King:** The meaning of a word is heavily dependent on the surrounding words and the real-world situation.
  - "The presentation was a bomb." (A failure) vs"That song is the bomb!" (Excellent)

The Core Challenge: Ambiguity

- "I saw a man on a hill with a telescope."
  - Did I use telescope?
  - Did the man have it?
  - Was the man on the hill or did I see from a hill?
- Ambiguity types:
  - **Lexical**
  - **Syntactic**
  - **Referential**
  - **Pragmatic**
- Reality:
  - Data is messy, context matters, and multiple parses exist

A High-Level View: The NLP Pipeline

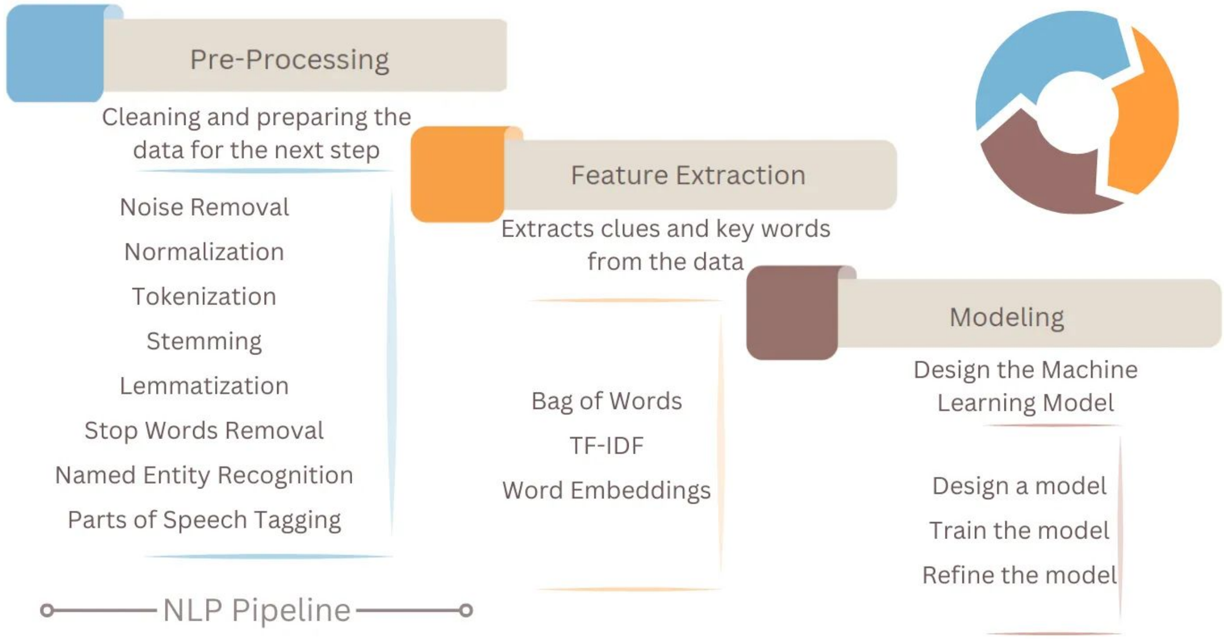
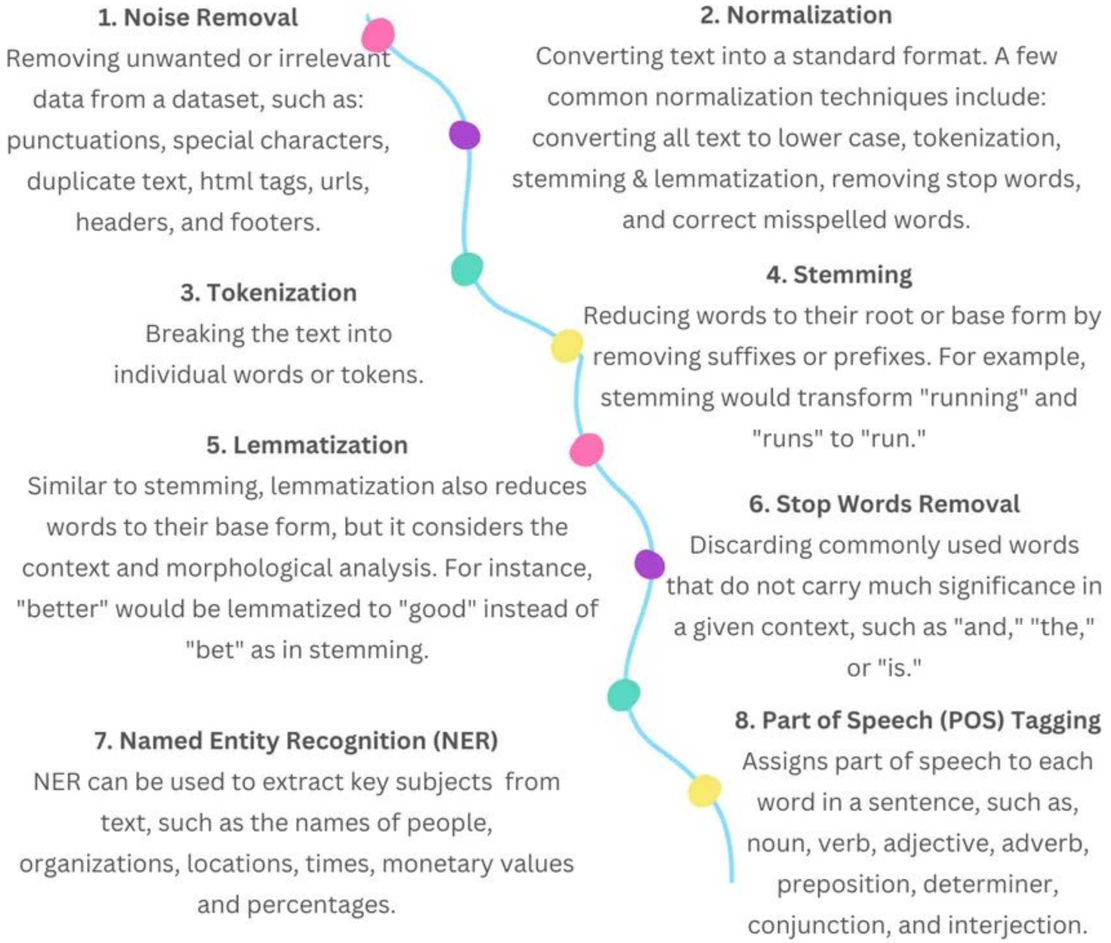


Figure: https://medium.com/@theaveragegal/natural-language-processing-nlp-pipeline-e766d832a1e5

Step 1: Text Preprocessing

- Raw Text
- Tokenization
- Normalization
- Stop Word Removal
- Stemming/Lemmatization
- POS Tagging



1. Noise Removal

Removing unwanted or irrelevant data from a dataset, such as: punctuations, special characters, duplicate text, html tags, urls, headers, and footers.

2. Normalization

Converting text into a standard format. A few common normalization techniques include: converting all text to lower case, tokenization, stemming & lemmatization, removing stop words, and correct misspelled words.

3. Tokenization

Breaking the text into individual words or tokens.

4. Stemming

Reducing words to their root or base form by removing suffixes or prefixes. For example, stemming would transform "running" and "runs" to "run."

5. Lemmatization

Similar to stemming, lemmatization also reduces words to their base form, but it considers the context and morphological analysis. For instance, "better" would be lemmatized to "good" instead of "bet" as in stemming.

6. Stop Words Removal

Discarding commonly used words that do not carry much significance in a given context, such as "and," "the," or "is."

7. Named Entity Recognition (NER)

NER can be used to extract key subjects from text, such as the names of people, organizations, locations, times, monetary values and percentages.

8. Part of Speech (POS) Tagging

Assigns part of speech to each word in a sentence, such as, noun, verb, adjective, adverb, preposition, determiner, conjunction, and interjection.

Figure: https://medium.com/@theaveragegal/natural-language-processing-nlp-pipeline-e766d832a1e5

Preprocessing: Tokenization

- Text: *"Natural Language Processing is fun!"*
- Tokens: ['Natural', 'Language', 'Processing', 'is', 'fun', '!']
- Tools: nltk.word_tokenize, spaCy tokenizer
- Why? Splits raw text into atomic units for further processing.

Demo: https://tiktokenizer.vercel.app/

Preprocessing: Normalization

- Text: *"NLP!!! is CoOl!*

- Normalized: "nlp is cool"

- Steps:
  a. Lowercasing
  b. Removing punctuation/emojis

- Goal: Make text consistent and reduce sparcity

Preprocessing: Stop Word Removal

Filtering Out the Noise

- Stop Words: Extremely common words that carry little semantic meaning in many NLP tasks.
    a. Examples: “a”, “an”, “the”, “in”, “of”, “and”, “to”
- **Why?** It can reduce the dimensionality of the data and improve model performance for tasks like topic modeling or text classification.
- **Caution:** Not always necessary. For tasks like machine translation or sentiment analysis where sentence structure matters, stop words can be important.

Preprocessing: Stemming

Chopping Words to their Root

- **Definition**: A crude, rule-based process of cutting off the beginning or end of words to get a common base form, known as the **stem**.
- **How it works:** Uses algorithms to chop off suffixes. The result may not be a real dictionary word.
- studies -> studi , studying -> studi
- **Pros:** Fast, simple, computationally inexpensive
- **Cons:** Can be inaccurate and produce non-words.

Figure:

Preprocessing: Lemmatization

Finding the Dictionary Form (Lemma)

- **Definition**: A more sophisticated process of reducing a word to its base or dictionary form, know as the **lemma**.
- **How it works:** Uses morphological analysis (understanding word structure) and a vocabulary/dictionary to determine the correct lemma.
- studies -> study , studying -> study
- **Pros:** More accurate and linguistically correct than stemming.
- **Cons:** Slower and requires more computational resources (like a dictionary).

Figure:

Stemming vs. Lemmatization

- **Stemming**: Fast, heuristic → sometimes non-words.
- **Lemmatization**: Uses linguistic rules → more accurate.

Word	**Stem**	**Lemma**
Studies	Studi	Study
Better	Better	Good
Running	Run	Run

Choosing Your Preprocessing Strategy: When and Why?

There is No Single Best Answer!

- **Scenario 1:** Search & Information Retrieval
  a. **Goal:** Match keywords efficiently
  b. **Strategy:** Use Stemming + Stop Word Removal
  c. **Why?** Speed is critical. We want to match "running", "runs" to the same core concept quickly. Stemming (**run**) does this efficiently. Removing stop words focuses the search on meaningful terms.

Figure:

Choosing Your Preprocessing Strategy: When and Why?

There is No Single Best Answer!

- **Scenario 2:** Sentiment Analysis
  a. **Goal:** Understand nuance and opinion
  b. **Strategy:** Use Lemmatization. Be careful with Stop Word Removal.
  c. **Why?** The difference between “good” and “better” is important, which lemmatization captures. Stop words like “**not**” completely invert the sentiment (e.g., “not good”). Removing them can destroy the meaning.

Figure:

Choosing Your Preprocessing Strategy: When and Why?

There is No Single Best Answer!

- **Scenario 3:** Machine Translation
  a. **Goal:** Preserve the full grammatical and semantic structure.
  b. **Strategy:** Minimal preprocessing. (Maybe just tokenization)
  c. **Why?** Every word, including stood words, contributes to the grammatical structure required for a correct translation. Models need the full, unaltered context.

Figure:

Is preprocessing still relevant today?

For Classic ML Models (TF-IDF, Naive Bayes):

- **YES, it's absolutely crucial.** These models have no prior knowledge of language. Preprocessing reduces the feature space, removes noise, and helps the model find the signal.

Figure:

Is preprocessing still relevant today?

For Modern Transformer Models (BERT, GPT, etc.):

- **The old rules DO NOT apply.**
- **DO NOT use Stemming/Lemmatization:** These models are pre-trained on massive amounts of raw text. They understand that "study" and "studying" are related but different. Removing this information can hurt performance.
- **DO NOT aggressively remove Stop Words:** Context is everything for Transformers. The position and presence of stop words are vital signals.
- **The "New" Preprocessing:** Transformers use their own sophisticated **Subword Tokenizers** (like WordPiece or BPE). These tokenizers break rare words into smaller, meaningful pieces (e.g., preprocessing -> pre, ##process, ##ing), which is a form of normalization that handles OOV (Out-of-Vocabulary) words gracefully.

Figure:

Code Example for Text Preprocessing

Figure:

Key Takeaways

- NLP has a rich history, evolving from rule-based systems to the powerful, Transformer-based LLMs of today.
- Preprocessing is the critical step of cleaning text, but it's **not one-size-fits-all**.
- The choice of technique (Stemming vs. Lemmatization, Stop Word Removal) **depends on your specific task and model.**
- Modern Transformer models require **minimal traditional preprocessing** because they learn from context and use their own advanced subword tokenizers.

Figure:

Next Class:

Text to Vectors & Word Embeddings