Questions & Answers on Chapter 01

❓ 1. List examples of real world applications of NLP

- **Email platforms** – e.g., Gmail spam detection
- **Voice assistants** – Siri, Alexa 🗣️
- **Search engines** – Google 🔍 understands queries
- **Machine translation** – Google Translate 🌍
- **Smart replies** – Messaging apps 💬
- **Grammar checkers** – Grammarly 📝
- **Chatbots** – Customer support 🤖
- **Healthcare** – Extract info from clinical notes 🏥

- Social media sentiment analysis

❓ 2. Explain the following NLP tasks: `language modelling`, `text classification`, `information extraction`, `information retrieval`, `conversational agent`, `text summarization`, `question answering`, `machine translation`, and `topic modelling`.

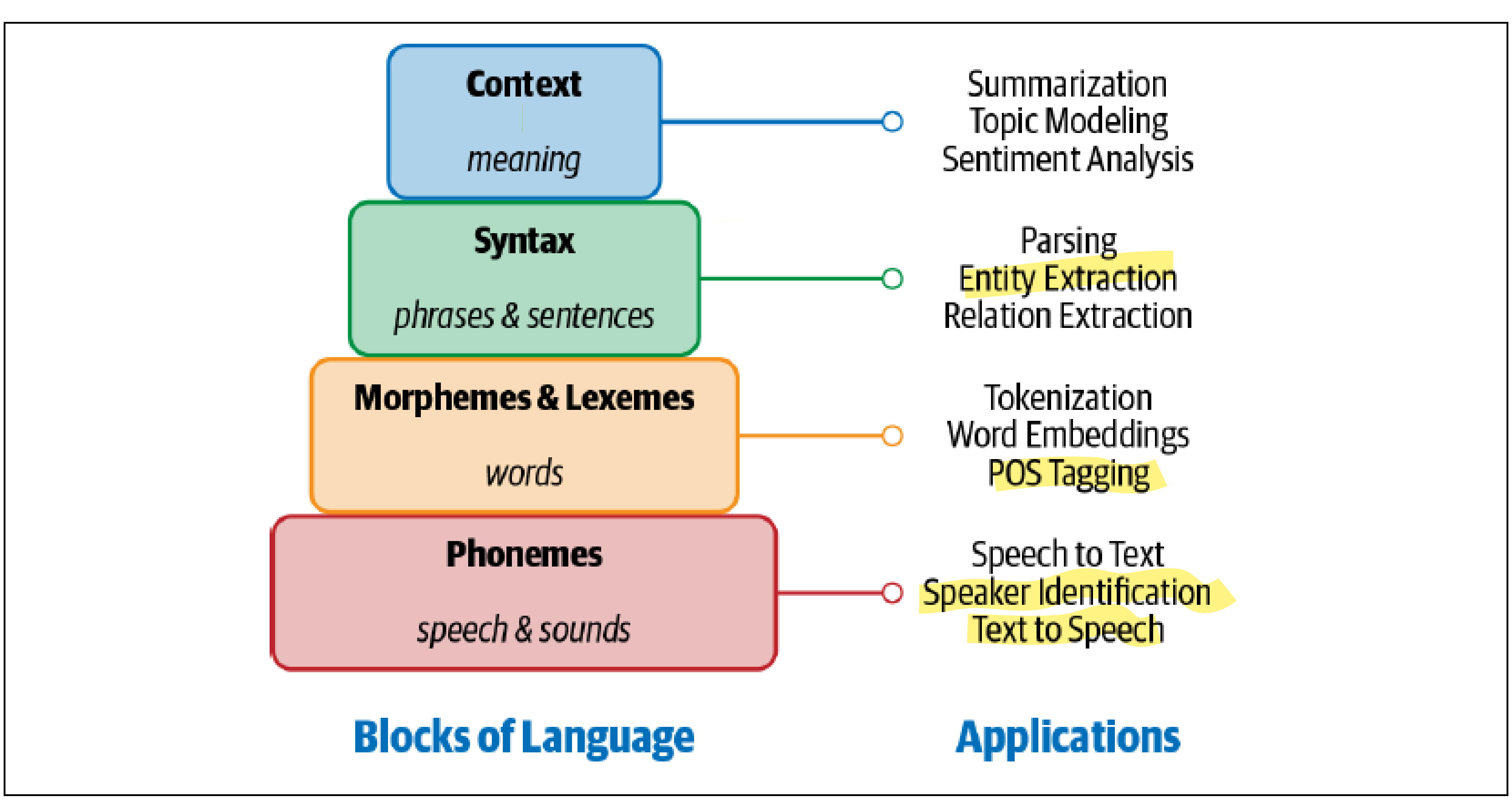
- 🔤 **Language Modelling**: Predict the next word in a sequence.
- 🗂️ **Text Classification**: Label text (e.g., spam vs non-spam).
- 📥 **Information Extraction**: Pull entities (e.g., names, dates).
- 🔎 **Information Retrieval**: Find relevant docs (e.g., Google).
- 🗣️ **Conversational Agent**: Systems that talk/understand (e.g., Siri).
- ✂️ **Text Summarization**: Shorten long texts.
- ❓ ➡️ ✅ **Question Answering**: Answer based on given input text. Watson AI
- 🌐 **Machine Translation**: Translate across languages.
- 📚 **Topic Modelling**: Find topics/themes in large text collections.

Group docs by themes

❓ 3. What are the building blocks of language and their applications?

Smallest sound units (e.g., "sh" in "shoe").

- 🔤 **Phonemes** – Smallest sound units → Used in speech-to-text
- 🧩 **Morphemes / Lexemes** – Meaningful units → Tokenization, embeddings
- 📐 **Syntax** – Sentence structure → Parsing & grammar checking
- 🌍 **Context** – Real-world meaning → NER, summarization



❓ 4. Why is NLP Challenging?

- 🤷 **Ambiguity** – Words have multiple meanings
- 🧠 **Lack of common sense** – Machines lack human knowledge
- 🎨 **Creativity** – Language is playful, ~~poetic~~, diverse Idioms, slang, poetry confuse models.
- 🌍 **Multilingual ~~complexity~~** – Different rules, scripts, dialects
  Diverse languages: Rules vary across languages.

❓ 5. How NLP, ML, and DL are related?

- 🧠 **AI** – The parent field
- 🤖 **ML** – Subfield: learns from data
- ⚡ **DL** – Subfield of ML: deep neural nets
- 🗣️ **NLP** – Uses ML/DL for understanding human language

❓ 6. Describe the heuristics-based NLP.

- 🛠️ Based on **rules or word counts**
  Count "positive" words to guess sentiment.
- Example: Sentence with "good" = positive 🙂
- ✅ Simple
- ❌ Not flexible, hard to scale

❓ 7. Explain briefly `Naive Bayes`, `Support Vector Machine`, `Hidden Markov Model`, And `Conditional Random Fields` approaches.

fast but assumes features are independent.
- 🧪 **Naive Bayes**: Probabilistic classification. Uses probability & Bayes' rule for classification.
- ⚙️ **SVM**: Draws a boundary between classes; great for high-dimensional data. robust but slow
  Models hidden states
- 🔒 **HMM**: Probabilistic model for sequences; great for speech tagging.
- 📝 **CRF**: Like HMM but more powerful; good for sequence labeling (NER, POS).
  Like HMM but better! Tags each word in context (e.g., "Apple" = company or fruit?).

❓ 8. What is the difference between RNN and LSTM NN?

- 🔁 **RNN**: Processes sequences; Remembers short-term steps.
- ⏳ **LSTM**: Advanced RNN that remembers **long-term context** better (uses gates 🧠 🚪)
  Remembers important info, forgets irrelevant details.

❓ 9. How CNN can be used for text processing?

- 🧱 Convert text to vectors
- 🧠 CNN identifies patterns (e.g., n-grams)
- ✅ Fast, great for classification tasks

❓ 10. Describe the concept transfer learning.

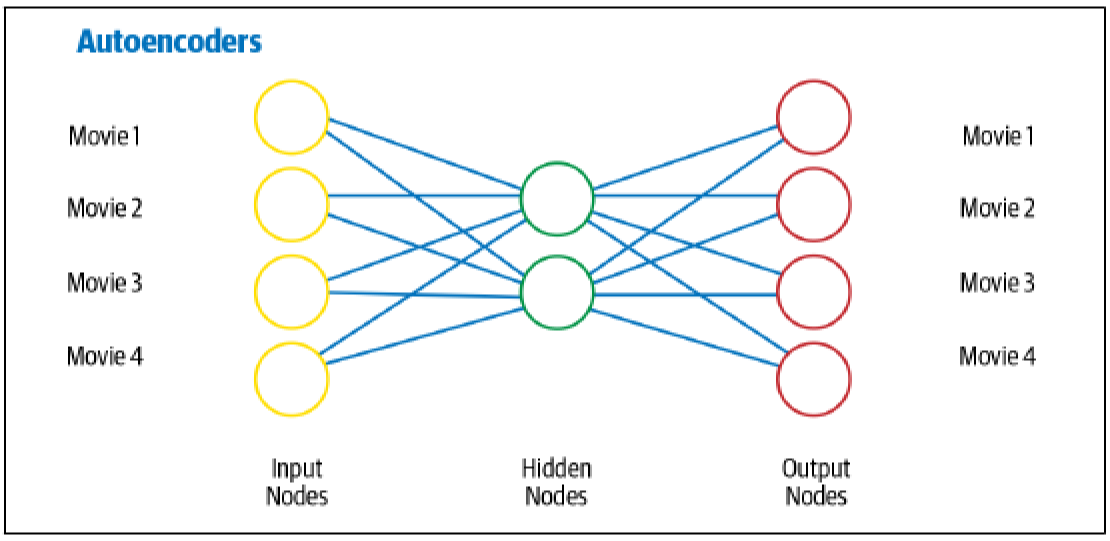
Remember Transformers

Pre-train on big data (e.g., BERT), then fine-tune for specific tasks.
- 📚 Train on one task → fine-tune for another
- Example: Pretrain on Wikipedia → Fine-tune for medical docs 💾
- ✅ Works well when labeled data is small

❓ 11. Give the architecture of autoencoder.

- ➡️ **Encoder** – Compress input
- 🔄 **Bottleneck** – Latent vector
- ⬅️ **Decoder** – Reconstructs original
- 🎯 Application: Feature learning, denoising

Input -> Encoder (compresses) -> Latent vector -> Decoder (reconstructs).

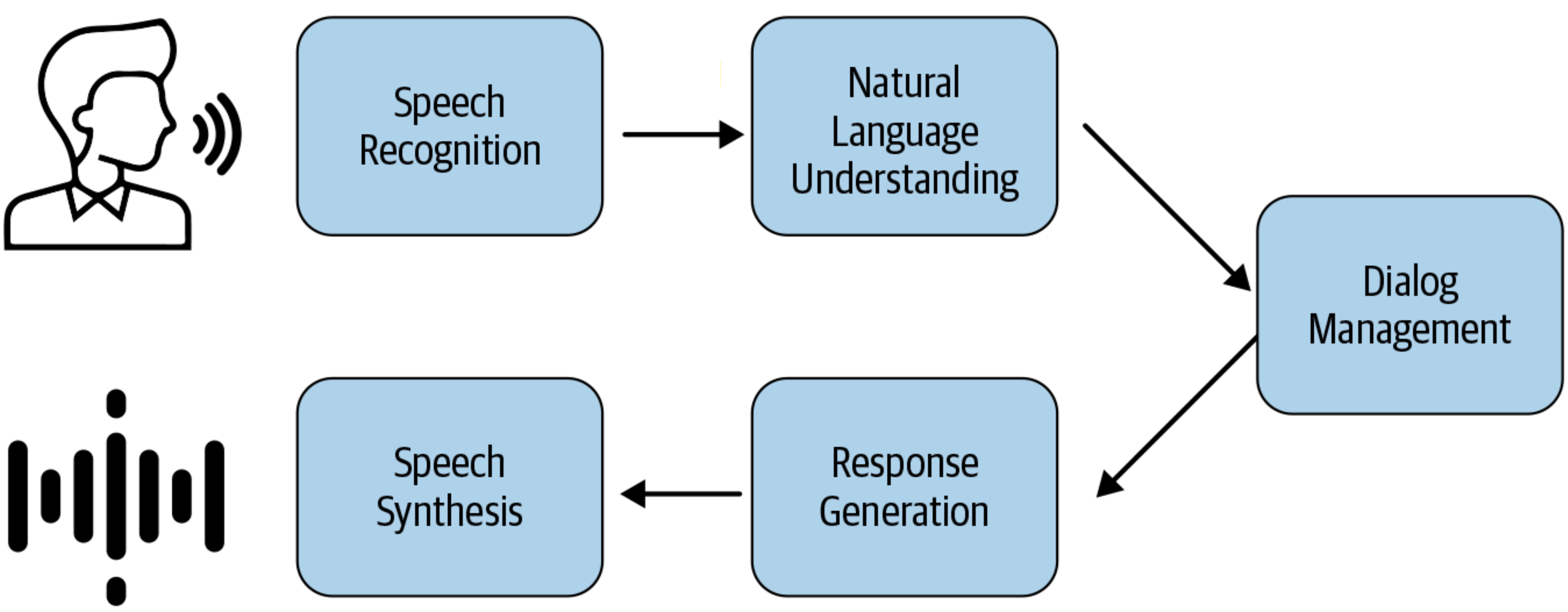


❓ 12. List the key reason that makes DL not suitable for all NLP tasks

- ⚠️ Needs lots of data → Overfitting on small sets
- 💸 Expensive to train/deploy
- 🧠 Hard to interpret
- 📦 Poor transferability to new domains   Fails with domain shifts
- 📱 Not efficient on low-power devices

❓ 13. Explain the flow of conversation agents

1. 🎤 **Speech Recognition**: Convert voice to text
2. 🧠 **Language Understanding**
   - Sentiment analysis 😕
   - Named Entity Recognition (NER) 🔍
   - Coreference resolution (who is "he"?)
3. 🧭 **Dialog Management** – Understand intent, decide next step
4. 💬 **Response Generation** – Generate a meaningful reply
5. 🎙️ **Text-to-Speech** – Speak the response (optional)



---

(56) إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ ۚ يَا أَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا

Questions & Answers on Chapter 03

❓ 1. List the four categories of text representation techniques.

- 🧱 **Basic vectorization**: One-hot, BoW, TF-IDF
- 🧠 **Distributed representations**: Word2Vec, GloVe
- 🌐 **Universal language representations**: BERT, ELMo
- 🛠️ **Handcrafted features**: Custom, domain-based

❓ 2. Describe the concept vector space models.

- 📏 A **Vector Space Model (VSM)** represents text as numerical vectors.
- 💡 These are inputs to ML models (e.g., for classification or similarity).

> 📌 *Example:*
> "Dog bites man" → `[1, 1, 1, 0, 0, 0]`

❓ 3. Use "D1: Dog bites man, D2: Man bites dog, D3: Dog eats meat, and D4: Man eats food" as an input, find their representation using one-hot encoding, bag of words, bag of N-gram, and TF-IDF.

TF (term-freq) = no.of term occurred in a doc / total no.of terms
IDF = log(total no.of docs / no.of docs contain the term)
TF X IDF

🔹 Vocabulary

First, build the vocabulary from all four docs:

1 2 3 4 5 6

Vocab: `[dog, bites, man, eats, meat, food]`

Size of vocabulary (|V|) = 6

🧩 1. One-Hot Encoding

Each word = a unique binary vector of size |V| = 6

Word	One-hot Vector
dog	[1, 0, 0, 0, 0, 0]
bites	[0, 1, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]
eats	[0, 0, 0, 1, 0, 0]
meat	[0, 0, 0, 0, 1, 0]
food	[0, 0, 0, 0, 0, 1]

Document representations (each word has its own row):

- D1 → dog = [1, 0, 0, 0, 0, 0] , bites = [0, 1, 0, 0, 0, 0] , man = [0, 0, 1, 0, 0, 0]

➡️ Simple but sparse, no semantic meaning, and can't handle unknown words (OOV).

🧺 2. Bag of Words (BoW)

We count how many times each word appears in a document. Order doesn't matter ❌📏

Doc	dog	bites	man	eats	meat	food
D1	1	1	1	0	0	0
D2	1	1	1	0	0	0
D3	1	0	0	1	1	0
D4	0	0	1	1	0	1

➡️ Pros: Easy, fixed-length, works well with simple models
➡️ Cons: No order, no context, high-dimensional

🔗 3. Bag of N-Grams (e.g., Bigrams = 2-grams)

Make sequences of 2 consecutive words:

- D1: `dog bites`, `bites man`
- D2: `man bites`, `bites dog`
- D3: `dog eats`, `eats meat`
- D4: `man eats`, `eats food`

Vocabulary of Bigrams: 2 4
[dog bites, bites man, man bites, bites dog, dog eats, eats meat, man eats, eats food]
1 3

Doc	dog bites	bites man	man bites	bites dog	dog eats	eats meat	man eats	eats food
D1	1	1	0	0	0	0	0	0
D2	0	0	1	1	0	0	0	0
D3	0	0	0	0	1	1	0	0
D4	0	0	0	0	0	0	1	1

➡️ Captures some **word order and context**, better than BoW
➡️ Still sparse and doesn't handle meaning

TF (term-freq) = no.of term occurred in a doc / total no.of terms in doc
IDF = log2(total no.of docs / no.of docs contain the term) = log2(N/DF)
DF -> Document Freq
TF X IDF

📊 4. TF-IDF (Term Frequency – Inverse Document Frequency)

TF = How often a word appears in a doc
IDF = How rare the word is across all docs

For example:

- "dog" appears in D1, D2, D3 → common → lower weight
- "meat" and "food" appear only once → higher weight

Resulting TF-IDF vector (simplified):

TF (Term Frequency)

Word	D1	D2	D3	D4
dog	0.33	0.33	0.33	0
bites	0.33	0.33	0	0
man	0.33	0.33	0	0.33
eats	0	0	0.33	0.33
meat	0	0	0.33	0
food	0	0	0	0.33

Word	TF-IDF (D1)
dog	low
bites	medium
man	low
eats	0
meat	0
food	0

IDF (Inverted Doc Frequency)

Word	DF	IDF
dog	3	0.41
bites	2	1
man	3	0.41
eats	2	1
meat	1	2
food	1	2

TF-IDF (TF(for word in each doc) X IDF)

Word	D1	D2	D3	D4
dog	0.33*0.41	0.136	0.136	0
bites	0.33	0.33	0	0
man	0.136	0.136	0	0.136
eats	0	0	0.33	0.33
meat	0	0	0.66	0
food	0	0	0	0.66

➡️ TF-IDF gives **importance** to rare words and downweights common ones
✅ Great for **document similarity** and **ranking**

❓ 4. Explain the difference between:

(a) distributional similarity and distributional hypothesis
(b) distributional representation and distributed representation

🔹 **(a)**

افتراض

- 📚 **Distributional hypothesis**: Words in similar contexts have similar meanings.
- 🔄 **Distributional similarity**: Measures context-based word similarity.

مقياس

🔹 **(b)**

ex: BOW, co-occurence matrix

- 🧱 **Distributional representation**: High-dim vectors from co-occurrence.
- 🧠 **Distributed representation**: Dense, learned vectors (e.g., Word2Vec, GloVe).

low-dimensional vectors (small)

❓ 5. Describe the wording embedding concept with an example of its use.

- 📦 **Word embeddings**: Dense vectors encoding meaning.
- ➡️ Similar words = close in vector space.

> 📌 *Example:*
> "USA" is close to "Canada", "Germany"

EX: In sentiment analysis, embeddings help models see that
"great" = "excellent" = "fantastic",
so different words with similar meaning give similar results.

❓ 6. Explain with an example the two architectural variants of Word2vec: CBOW and SkipGram.

- 🧩 **CBOW**: Predict center word from context
- 🧠 **SkipGram**: Predict context from center word

> 📌 *Example:*
> Sentence: “The cat sat on the mat”

- CBOW: ("The", "sat") → "cat"
- SkipGram: "cat" → ("The", "sat")

Out-of-Vocabulary

❓ 7. How the OOV problem can be solved?

- 🛠️ Subword models (prefix/suffix)
- 🔤 Character-level or byte-level models
- ⚙️ Random init for unknown words
- 💬 Use **fastText** (subword-based vectors)

❓ 8. What is the difference between Doc2vec and Word2vec?

- 🧠 **Word2Vec**: Word-level vectors
- 📄 **Doc2Vec**: Document-level vectors
  ➡️ Doc2Vec = Word2Vec + document context

❓ 9. What are the important aspects to keep in mind while using word embeddings?

تحيز

- ⚠️ Bias can leak from data
- 📦 Embeddings = large file sizes
- 💡 Not all linguistic structure is captured
- 🧠 Might need extra sentence/document-level features

❓ 10. How high-dimensional data can be represented visually?

- 📉 Use **t-SNE**:
  - Reduces high-dim embeddings (e.g., 300D) → 2D/3D
  - Useful for visualizing clusters/semantic similarity

[illegible] Cluster

❓ 11. With example explain the use of handcrafted feature representations.

- 🧠 Example: **TextEvaluator** (by ETS)
- 📚 Used for estimating reading difficulty
- Handcrafted features include:

- Syntactic complexity جملة معقدة
- Word familiarity
- Concreteness و محسوس

> Useful for educational and grading tasks.

إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ ۚ يَا أَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا (56)

Questions & Answers on Chapter 04

❓ 1. What is the difference between binary, multi-class, and multi-label classification?

- **Binary**: Two classes only (e.g., spam vs. not spam)
- **Multiclass**: More than two classes, choose one (e.g., positive / neutral / negative)
- **Multilabel**: One text can have multiple labels (e.g., news tagged as "sports" and "politics")

❓ 2. Give some applications of text classification.

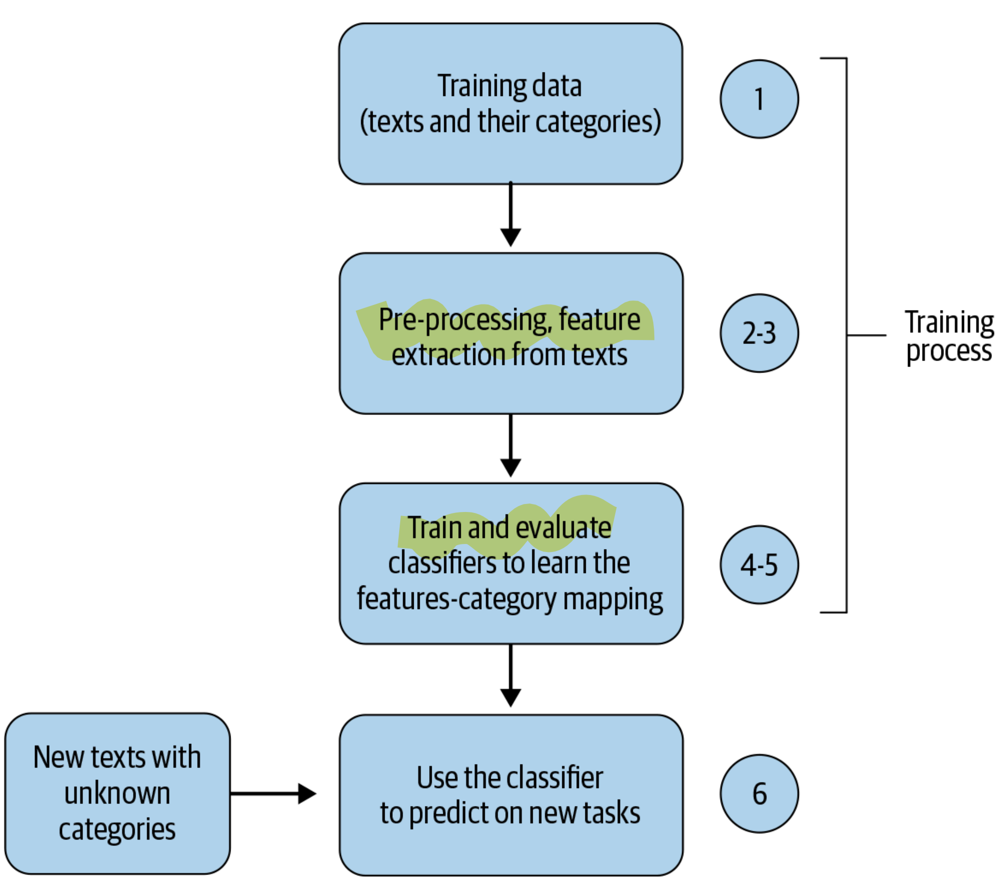
Classifying articles, blogs, or news into topics (e.g., sports, tech)

- 🗂️ Organizing content (e.g., news, blogs)
- 🙋 Customer support automation
- 🛒 Sentiment analysis in e-commerce Detecting positive/negative reviews about products
- 🈯 Language identification Detecting the language of the text (English, Arabic, French...)
- 📰 Fake news detection Identifying whether a news article is real or fake

- Spam filtering -> Classifying emails as "Spam" or "Not Spam"

❓ 3. Describe the pipeline for building text classification systems.

1. Collect labeled data
2. Split into train/test/validation sets
3. Convert text to features (vectors)
4. Train the model 🧠
5. Evaluate using metrics (accuracy, F1, etc.)
6. Deploy the model 🚀 and monitor performance



❓ 4. Classification can be done without the text classification pipeline, explain how?

dictionary

You can use a **lexicon-based approach**:

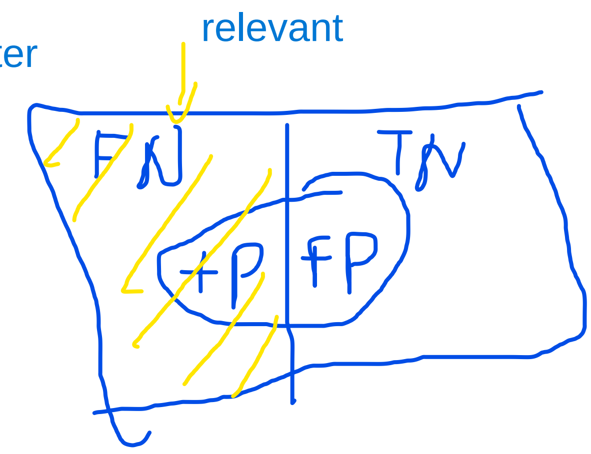
- Make lists of positive and negative words
- Use rules (e.g., 😄 = positive, 😡 = negative)

> Simple, but limited and not scalable for complex tasks.

❓ 5. Describe with an example the confusion matrix of a classifier.

A **confusion matrix** helps measure performance by comparing actual vs predicted labels.

Example (binary classification): ex: Spam Filter



	Predicted 👍	Predicted 👎
Actual 👍	True Pos (TP)	False Neg (FN)
Actual 👎	False Pos (FP)	True Neg (TN)

Metrics:

- **Precision** = TP / (TP + FP) tp / retrieved (tp+fp)
- **Recall** = TP / (TP + FN) tp / relevant (tp+fn)

❓ 6. List the potential reasons for poor classifier performance.

- 😕 Wrong or weak algorithm The chosen model is too simple or not suitable
- 🔍 Poor pre-processing / features Text not cleaned properly, or bad feature extraction
- ⚙️ Bad hyperparameter tuning Parameters like learning rate, tree depth, etc., are not optimized
- 📉 Imbalanced data or noisy labels One class has way more data than others
  - Noisy or incorrect labels Training data contains wrong or inconsistent labels
  - Too little training data

❓ 7. How to solve class imbalance problem of a dataset?

- ⚖️ Use balanced metrics (e.g., F1 score) Instead of accuracy, use F1-score, precision, recall
- 🔁 Resample data (oversample or undersample) SMOTE
- 🎯 Try class-weighted models give more weight to the minority class
- 🧠 Train custom models that adjust for imbalance

❓ 8. What is the difference between generative and discriminative classifiers?

- 🎲 **Generative**: Models how data is generated → calculates P(x|y) and P(y) (e.g., Naive Bayes) Generator => models the world
- ⚖️ **Discriminative**: Directly models decision boundary → learns P(y|x) (e.g., Logistic Regression, SVM) Decision => draws the line between classes

---

Dense vectors encoding meaning. Similar words = closer in vector space

❓ 9. How to use word embeddings as features for text classification?

1️⃣ Replace words with **pre-trained vectors** (Word2Vec, GloVe) each word replaced with vector
2️⃣ Average them or feed them into a neural net convert to numeric representation
✅ Captures meaning and similarity → using model
📌 Used for tasks like **sentiment classification**

❓ 10. List the steps for converting training and test data into a format suitable for the neural network.

1. Tokenize the sentences 🧩 Converts text into machine-readable words/tokens.
2. Pad them to same length 📏 [1, 2, 3] -> [1, 2, 3, 0] (with padding)
3. Map words to embedding vectors Represent words as meaningful dense vectors.
4. Feed embeddings into neural net (CNN, LSTM, etc.) Learns patterns and performs classification.

❓ 11. Which technique is better for text classification CNN or LSTM and why?

- ✅ Use **LSTM** for long sequences or large datasets (remembers order)
- ⚡ Use **CNN** for shorter text and faster training
- 🎯 LSTM = better for capturing long-term dependencies
- ⚙️ CNN = good at local patterns (like n-grams)

Use LSTM when word order matters or text is long.
Use CNN when you need speed and are okay with losing some sequence info.

❓ 12. How text classification models can be interpreted?

Use **LIME** 🧠 (Local Interpretable Model-Agnostic Explanation):

- Explains predictions word by word
- Shows which features affected the decision

✅ Useful in sensitive tasks (e.g., medical, legal)
  Medical predictions
  Legal document classification
  Any critical decision-making task

Example:
If a classifier predicts a review is positive, LIME might highlight:
"I love this phone" => LIME shows "love" had the biggest impact

---

❓ 13. How to solve no training and less training data problems?

- 📉 **No data** → Create dataset by annotation — manual annotation (label the data yourself or with help)
- 📊 **Less data** → Use:
  - **Active learning** Let the model choose the most useful examples to label
  - **Domain adaptation** Use models trained on similar tasks/domains
  - **Weak supervision** (rules or heuristics) Use rules, patterns, or heuristics to auto-label data

---

❓ 14. Give some options to explore when no labels exist for a dataset.

- 🧰 Use pre-built APIs (Google NLP, etc.)
- 📚 Find public labeled datasets
- 🧪 Apply weak supervision
- 🤖 Try active learning
- 👍 Learn from feedback over time

❓ 15. Describe the pipeline for building a classifier when there is no training data.

1. **Define rules/heuristics** "Love" -> pos, "bad" -> neg , ...
2. Use **Snorkel** or other tools for weak labeling
3. Leverage **existing models/APIs** (Google NLP – HuggingFace models)
4. Apply **active learning** to label selectively
5. Iterate and refine with **real user feedback**

إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ ۚ يَا أَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا (56)

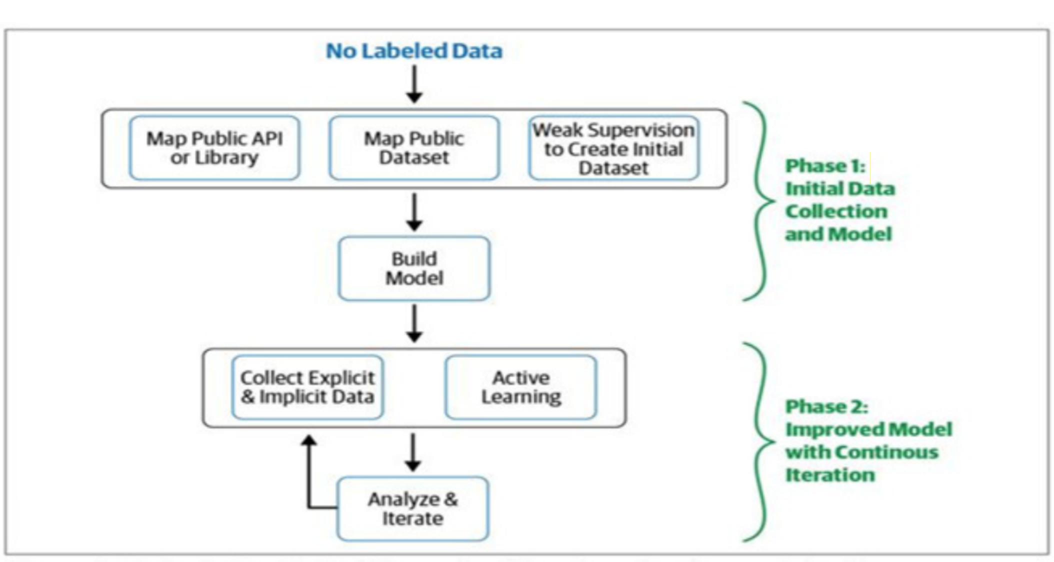
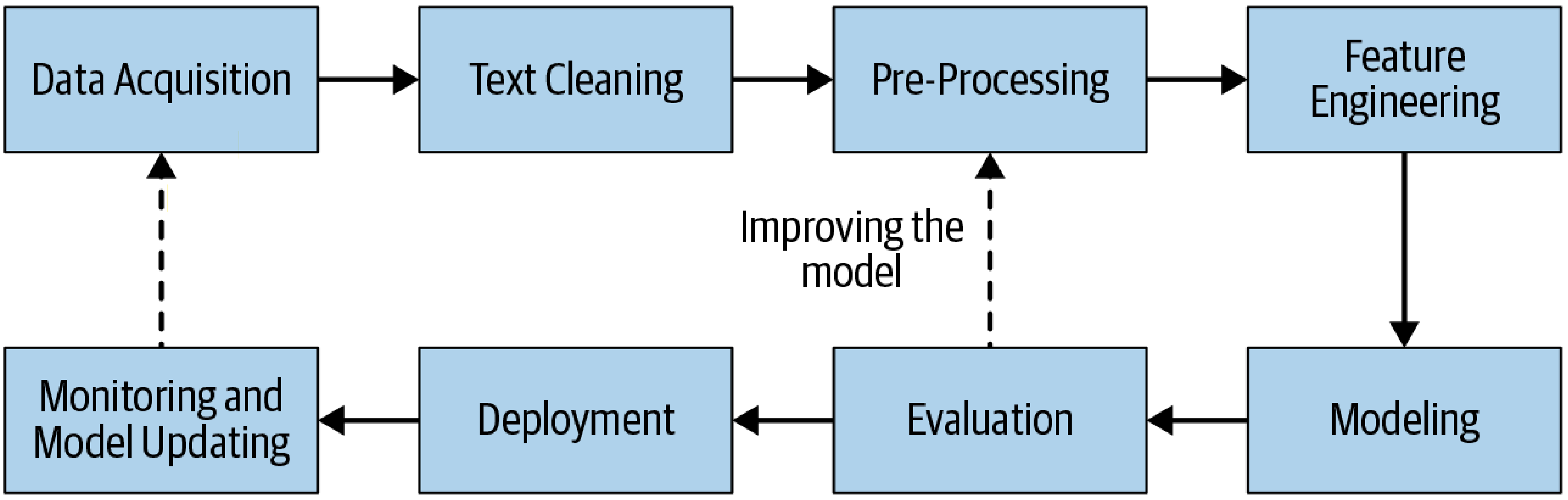


Figure 4-11. A pipeline for building a classifier when there's no training data

Questions & Answers on Chapter 02

❓ 1. What are the key stages of a generic pipeline for NLP system development?

- 🧺 **Data acquisition** → Get the data
- 🧼 **Text cleaning** → Remove noise like HTML, symbols
- 🔤 **Pre-processing** → Tokenize, lowercase, etc.
- 🧬 **Feature engineering** → Convert text into numbers
- 🧠 **Modeling** → Train the ML/DL model
- 🎯 **Evaluation** → Test performance with metrics
- 🚀 **Deployment** → Integrate into a product
- 📈 **Monitoring & Updating** → Improve with feedback



❓ 2. How can we get data required for training an NLP technique?

- ✅ Use **public datasets** (Kaggle, Hugging Face)
- 🕷️ **Scrape websites** intervention
- 📲 **Product instrumentation** (collect from user input)
- 🔁 **Data augmentation**:
  - 🔄 Synonym replacement ("happy" => "joyful")
  - 🌍 Back translation (Translate EN => FR => EN)
  - 🔁 Bigram flipping ("machine learning" => "learning machine")
  - 👨‍🌾 Entity replacement
  - 🙊 Add noise ("hello" => "hlelo")

❓ 3. List the different data augmentation methods

- 🔄 **Synonym replacement** – Swap words with similar ones
- 🌍 **Back translation** – Translate text to another language then back
- ↔ **Bigram flipping** – Swap pairs of words
- 👨‍🎓 **Entity replacement** – Change names/places with others
- 📢 **Add noise** – Introduce typos or symbols randomly
- 🧪 **Tools**:
  - 🌀 **Snorkel** – Weak supervision
  - ⚙️ **EDA**, **NLPAug** – Generate synthetic data
  - 🎯 **Active learning** – Label the most useful samples

❓ 4. Data can be collected from PDF files, HTML pages, and images. How can this data be cleaned based on their sources?

Use PyPDF2 to extract text.
- 📄 **PDFs** → Use PyPDF/Text extraction

Remove tags
- 🌐 **HTML** → Strip tags using BeautifulSoup

Extract text via
- 🖼️ **Images** → Use OCR (e.g., Tesseract)
- 👉 **Then**: remove formatting, special characters, fix encodings

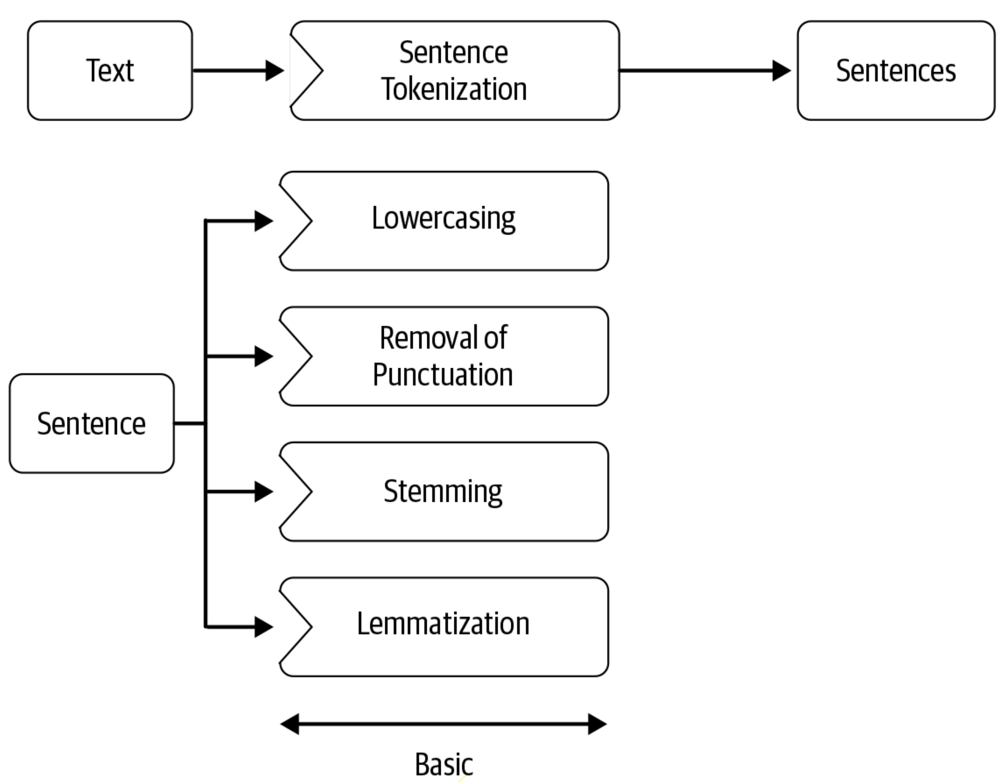
---

❓ 5. Using dot (.) to segment sentences can cause problems, explain how?

- **Problem**: Splits abbreviations (e.g., "Mr. Smith" → "Mr" + ".").
- **Solution**: Use NLP libraries (spaCy, NLTK) to detect context.

❓ 6. What are the frequent steps in the data pre-processing phase?

1. Stop word removal 🚫 ("the", "is").
2. Stemming/Lemmatization 🌱 ("running" → "run").
3. Remove digits/punctuation 🔢 ("Price: $100" → "price").
4. Lowercasing 🔠 ("Hello" → "hello").



❓ 7. With examples, explain the differences between segmentation and lemmatization

Splitting text into sentences/words

📏 **Segmentation** – Break text into parts
→ *"Hi. I'm Anas."* → 2 sentences

📚 **Lemmatization** – Reduce to base form
→ *"Was" → "be", "Better" → "good"*

❓ 8. What is the difference between code mixing and transliteration?

Mixing languages in one sentence

🌐 **Code mixing** – Switching languages mid-sentence
→ *"Ana hungry w 3ayez akol"*

🔡 **Transliteration** – Non-English written using English letters
→ *"shukran" instead of شكراً*

❓ 9. Describe the concept of coreference resolution

Link pronouns to their nouns.

🔁 **Coreference resolution** links words to the same entity
→ *"Sara went home. She was tired."* → "She" = "Sara"
📌 Used in summarization, QA, info extraction

❓ 10. Explain the feature engineering for classical NLP vs DL-based NLP

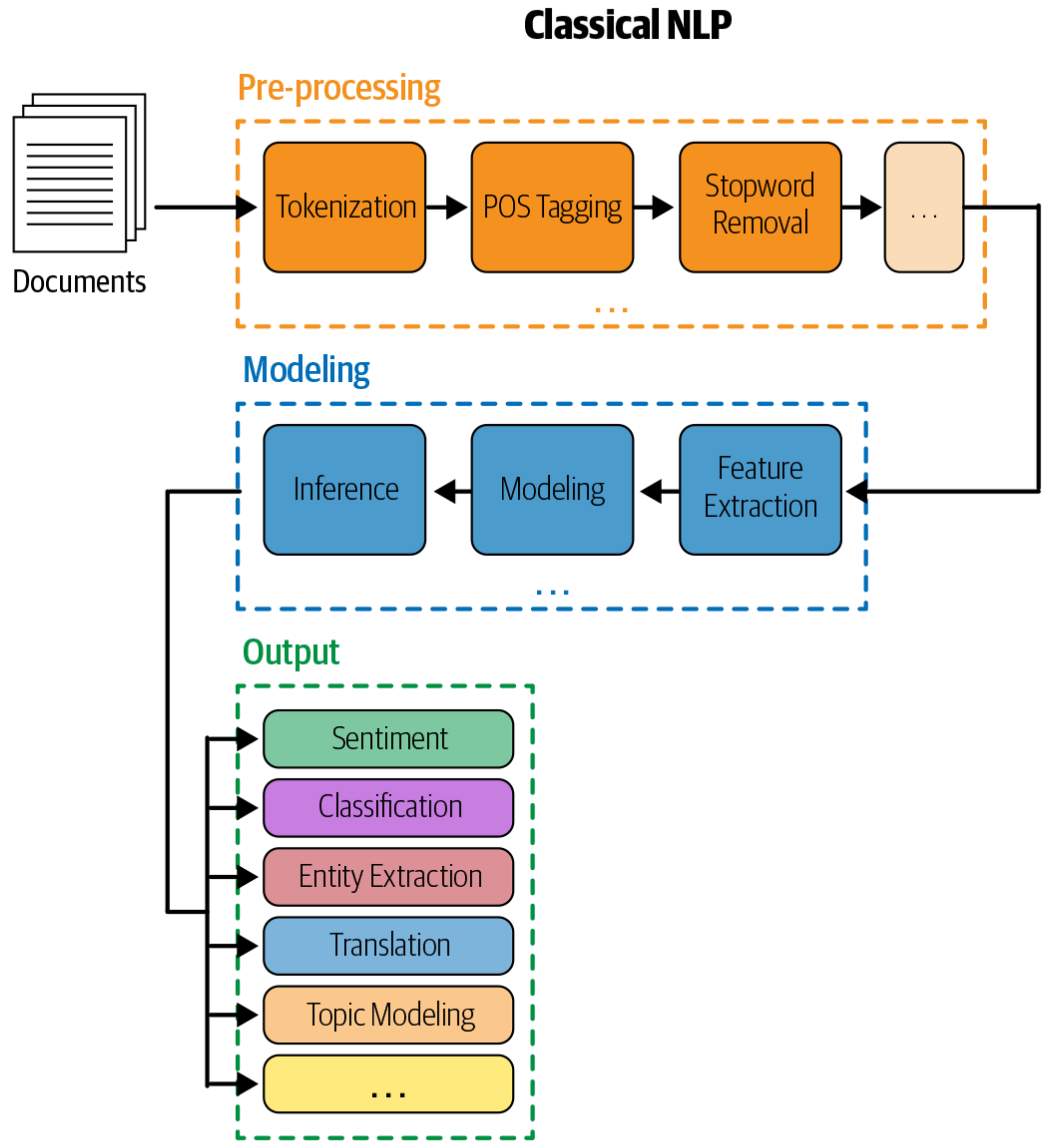
🛠️ **Classical NLP**: Example: Counting "positive" words for sentiment analysis

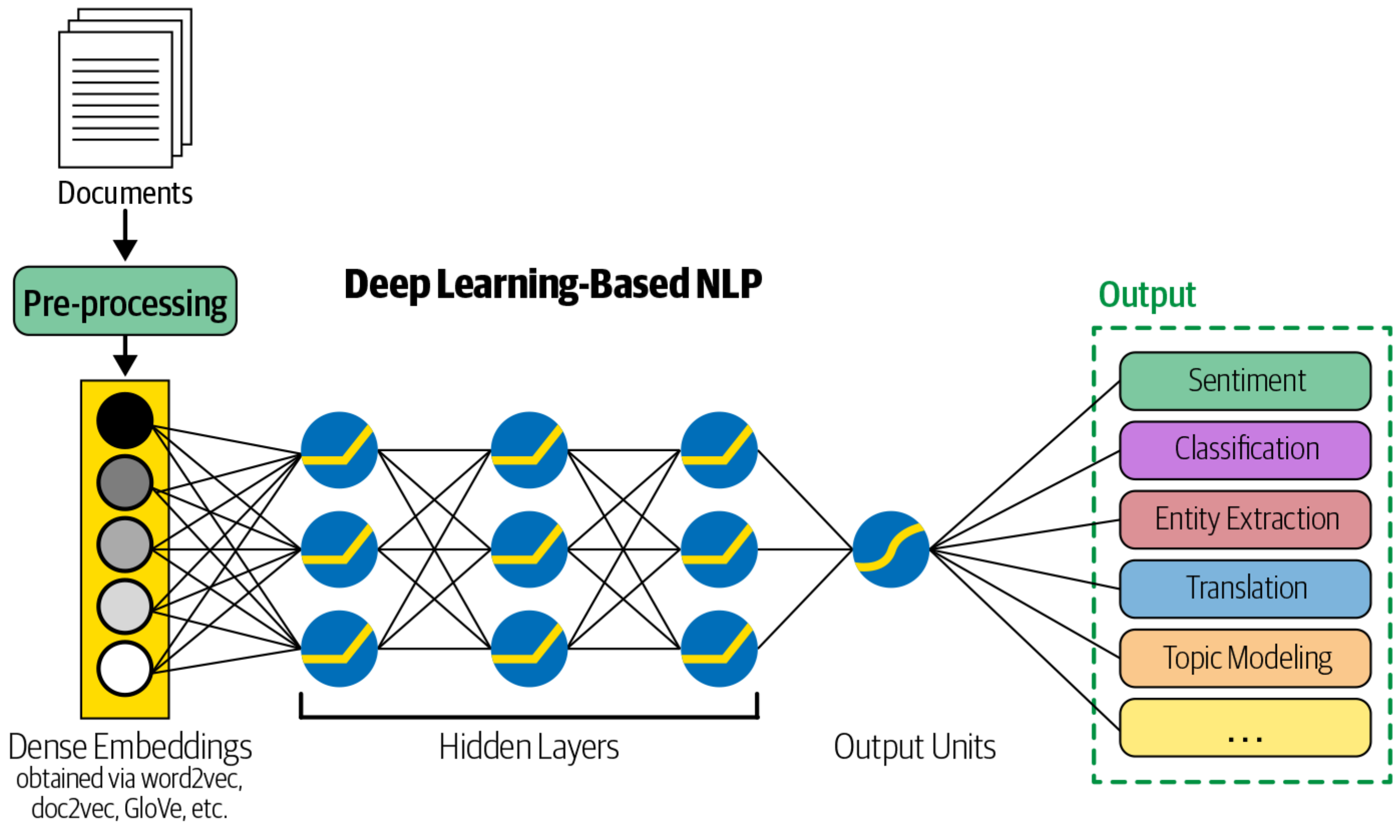
Handcrafted features

- Manual features (BoW, TF-IDF, sentiment, etc.)

⚡ **Deep Learning**: (BERT embeddings)

- Raw text → embeddings → automatic feature learning





❓ 11. How to combine heuristics directly or indirectly with the ML model?

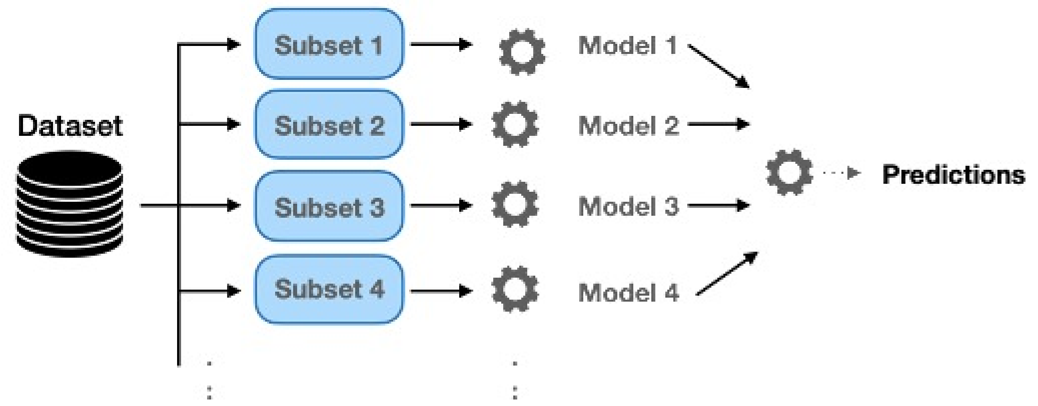
- **As features** – Add them to model input
- **Preprocessing** – Use to filter or convert data

✅ Best when heuristics are reliable!

❓ 12. What is the difference between model ensembling and stacking?

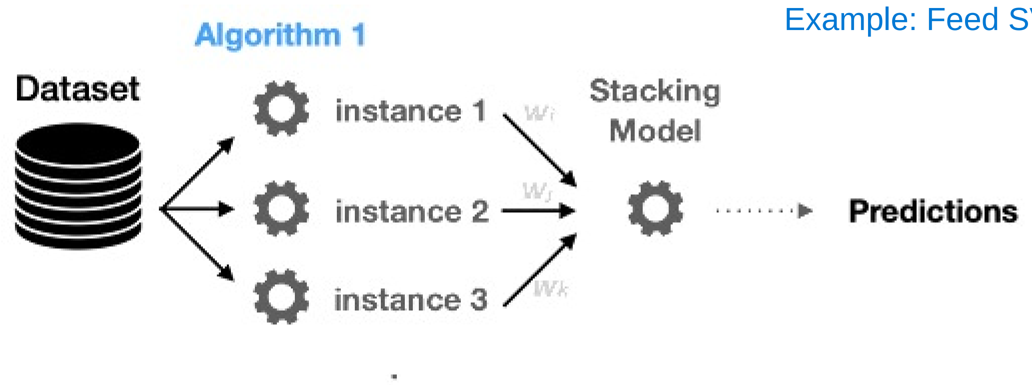
from multiple models

📊 **Ensembling** – Combine predictions (average, majority vote)



📦 **Stacking** – Use one model's output as input to another

Example: Feed SVM predictions into a logistic regression model



❓ 13. Which modeling technique can be used in the following cases: `small data`, `large data`, `poor data quality`, and `good data quality`?

- 📉 **Small data** → Rule-based or traditional ML
- 📈 **Large data** → Deep learning
- 🧽 **Poor data** → Clean more first
- 🌟 **Good data** → Use off-the-shelf models quickly
  Use cloud APIs (e.g., Google NLP).

❓ 14. What is the difference between intrinsic and extrinsic evaluation?

numbers

📊 **Intrinsic** – Direct metrics (e.g., accuracy, F1)
🎯 **Extrinsic** – Impact on task (e.g., user satisfaction)
, resolution time

❓ 15. What are the metrics that can be used in: `classification`, `measuring model quality`, `information retrieval`, `predication`, `machine translation`, and `summarization tasks`? 🎯

- 🧪 **Classification** → Accuracy, F1- score
- 📊 **Model quality** → AUC   Confusion Matrix
- 🔎 **Info retrieval** → MRR, MAP
- 📈 **Prediction** → RMSE (for regression)
- 🌐 **Machine translation** → BLEU, METEOR
- ✂️ **Summarization** → ROUGE

❓ 16. Describe `deploying`, `monitoring`, and `updating` phases of NLP pipeline

🚀 **Deployment** – Expose model as service/API
📉 **Monitoring** – Track feedback, quality drift   Track accuracy drops
🔄 **Updating** – Periodically retrain on new data

❓ 17. Explain how the NLP pipeline is different from one language to another

🌍 **English** – Easier segmentation, fewer morphology issues
🈷️ **Chinese/Arabic** – Need custom tokenizers, more complex grammar
🧠 Different languages may require **different tools, steps, and models**

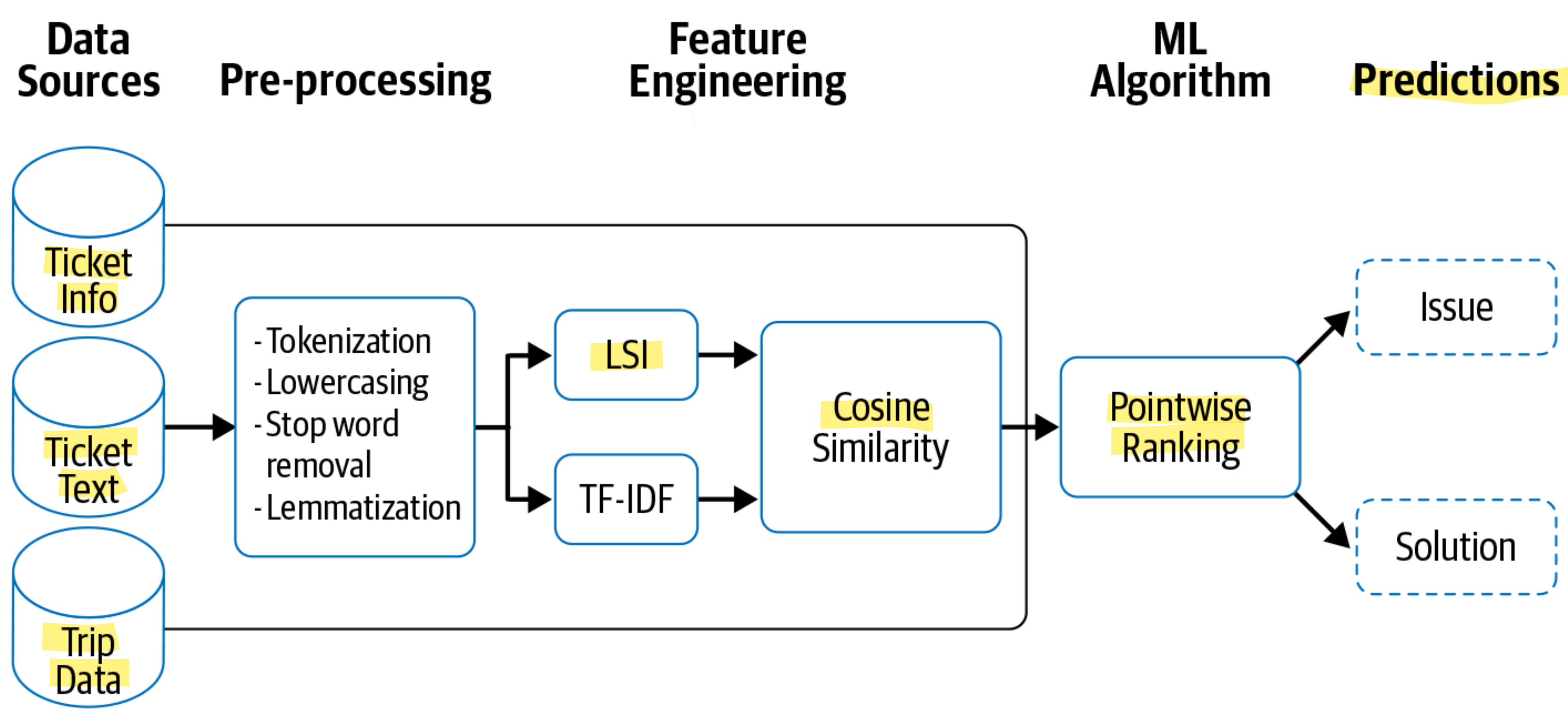
❓ 18. Describe the NLP pipeline for ranking tickets in a ticketing system by Uber

📥 **Collect** ticket → Clean HTML
🧹 **Preprocess** – tokenize, lowercase, remove stop words, lemmatize
📚 **Feature engineering**:

- BoW, TF-IDF, LSI Topic Modeling
- Compare ticket & topics with **cosine similarity**

Training using model -> train data = ticket + trip data
🧠 **Modeling** – Use ticket + trip data
🎯 **Ranking** – Select top 3 solutions
📈 **Evaluation**:

- 📊 **Intrinsic** – MRR
- 🎯 **Extrinsic** – Customer satisfaction, resolution time



إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ ۚ يَا أَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا (56)