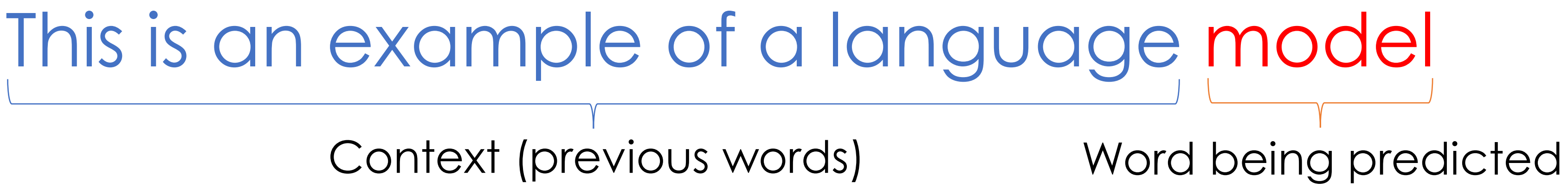
CHAPTER 6

Language Modeling & NLP Evaluation

- Language modeling
- N-gram language model
- NLP System Evaluation
  - Classification
    - Confusion matrix, Accuracy, Recall, Precision, F1-Score
  - Text generation
    - ROUGE

Language model?

Language Modeling is a technique used in NLP that involves predicting the next word in a sentence or sequence of words based on the context and previous words.



This is an example of a language model

Context (previous words)

Word being predicted

Types:

- Statistical language models
- Neural language models

N-gram language model

An **n-gram model** is a probabilistic language model that **predicts** the **likelihood of a word** based on the **previous n−1 words** in a sequence.

Note: The probability of a word depends only on a fixed, **limited context** (the n-1 preceding words), not the entire sentence.

Applications:

- Text generation,
- Speech recognition,
- Machine translation..

N-gram language model

- **N-gram**: A sequence of **n** consecutive words (e.g., bigram = 2 words, trigram = 3 words)
- **Probability:** the model estimates the probability of generating the **i^th^** word from **n-1** previous words

$$P\left(w_i \middle| w_{i-(n-1)}, \ldots w_{i-1}\right) = \frac{Count(w_{i-(n-1)}, \ldots w_{i-1}, w_i)}{Count(w_{i-(n-1)}, \ldots w_{i-1})}$$

Example: predict the next word of “He loves”

$$P("cats"|"He\ loves") = \frac{Count("He\ loves\ cats")}{Count("He\ loves")}$$

N-gram language model

Training: Learn the probabilities of n-grams from the training corpus

- Count the occurrences of all possible n-grams in the dataset
- Compute conditional probabilities for each n-gram

Example: 3-gram model

3-grams	Count 3-grams in Corpus	Output
(« Model », « of », « language »)	5	P(« language »| «Model of ») = 5/200 = 0,025
(« Model », « of », « Markov »)	25	P(« Markov »| «Model of ») = 25/200 = 0,125
(« Model », « of », « network »)	10	P(« network »| «Model of ») = 10/200 = 0,05
…	…	…
(« Model », « of », *)	200	

Predict the next word in the context "Model of":

- Model of **Markov** (Highest probability)

N-gram language model

- **Limitations:**
  - Cannot handle long-range dependencies (limited context).
  - Struggles with unseen n-grams (zero probabilities for OOV words)

3-grams	Count 3-grams in Corpus	Output
(« Model », « of », « language »)	5	P(« language »| «Model of ») = 5/200 = 0,025
(« Model », « of », « Markov »)	25	P(« Markov »| «Model of ») = 25/200 = 0,125
(« Model », « of », « network »)	10	P(« network »| «Model of ») = 10/200 = 0,05
…	…	…
(« Model », « of », *)	200	

If next word in the context "Model of" is "Bayes":

- P(« Bayes »| «Model of ») = 0

N-gram language model

- **Smoothing:**
  - Rare or unseen words (OOV) are replaced with “UNK” token
  - Use a smoothing rate δ

3-grams	Count 3-grams in Corpus	Output
(« Model », « of », « language »)	5	P(« language »| «Model of ») = 5/200 = 0,025
(« Model », « of », « Markov »)	25	P(« Markov »| «Model of ») = 25/200 = 0,125
(« Model », « of », « network »)	10	P(« network »| «Model of ») = 10/200 = 0,05
(« Model », « of », « UNK»)	0	P(« UNK »| «Model of ») = 0
…	…	…
(« Model », « of », *)	200	

If next word in the context “Model of” is “Bayes”:

- Replace “Bayes” with “UNK”:
- P(« Bayes »| «Model of ») = 0

N-gram language model

- **Smoothing:**
  - Rare or unseen words (OOV) are replaced with “UNK” token
  - Use a smoothing rate δ

$$P\left(w_i \middle| w_{i-(n-1)}, \dots w_{i-1}\right) = \frac{\delta + Count(w_{i-(n-1)}, \dots w_{i-1}, w_i)}{\delta(|V| + 1) + Count(w_{i-(n-1)}, \dots w_{i-1})}$$

  - $|V|$ is the vocabulary size

N-gram language model

- **Smoothing:**
  - Smoothing $\delta = 0.1$ $|V| = 999$

3-grams	Count 3-grams	Output
(« Model », « of », « language »)	5	P(« language »| «Model of ») = (0.1+5)/(100+200) = 5.1/300 = 0,017
(« Model », « of », « Markov »)	25	P(« Markov »| «Model of ») = (0.1+25)/(100+200) = 25.1/300 = 0,083
(« Model », « of », « network »)	10	P(« network »| «Model of ») = (0.1+10)/(100+200) = 10.1/300 = 0,033
(« Model », « of », « Bayes»)	0	P(« Bayes »| «Model of ») = (0.1+0)/(100+200) = 0.1/300 = 0,0003
…	…	…
(« Model », « of », *)	200	

$$P\left(w_i \middle| w_{i-(n-1)}, \dots w_{i-1}\right) = \frac{\delta + Count(w_{i-(n-1)}, \dots w_{i-1}, w_i)}{\delta(|V| + 1) + Count(w_{i-(n-1)}, \dots w_{i-1})}$$

NLP system evaluation

- Evaluating an NLP system is a critical process to ensure it meets its intended purpose and performs effectively.

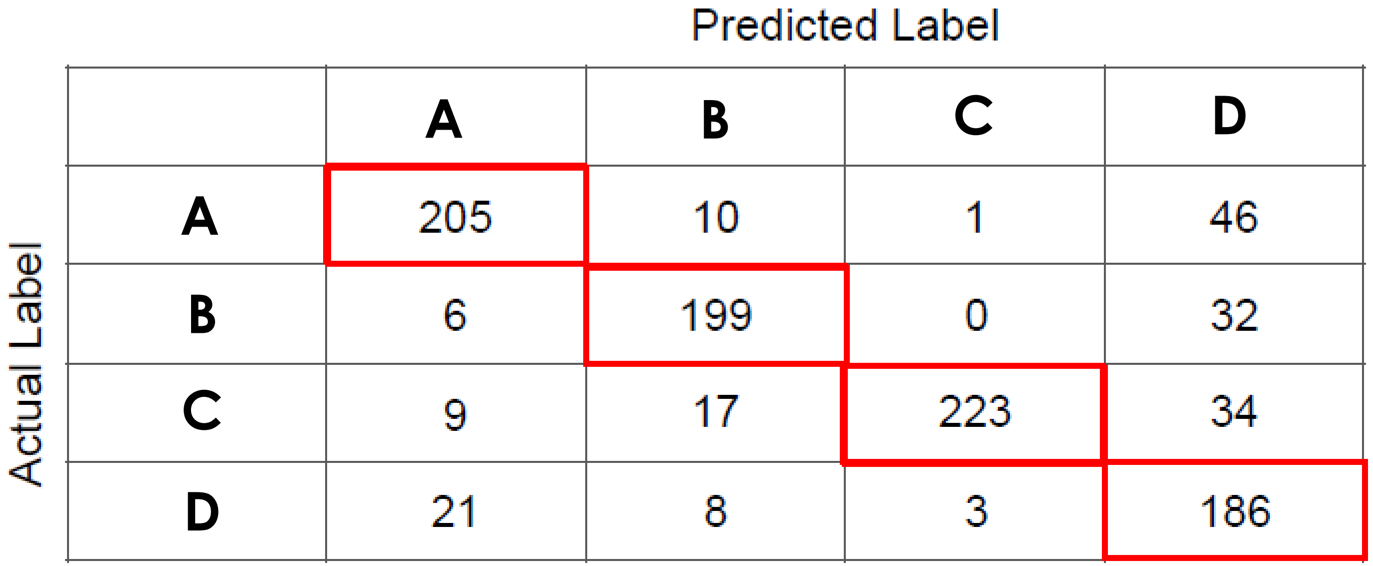
- **Evaluation methods** depend on the specific tasks the system is designed for, such as text classification, machine translation,..

Classification

Evaluation

Confusion matrix

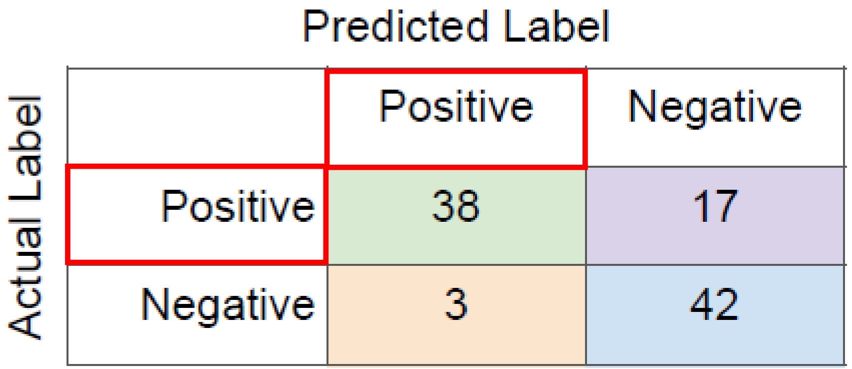
- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



		Predicted Label			
		A	**B**	**C**	**D**
Actual Label	**A**	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

$$Total\ accuracy = \frac{\sum correct}{\sum all} = \frac{813}{1000} = 0.813$$

Confusion matrix



		Predicted Label	
Actual Label		Positive	Negative
	Positive	38	17
	Negative	3	42

True Positive (TP): Predicted positive matches actual positive

True Negative (TN): Predicted negative matches actual negative

False Positive (FP) (“Type I Error”): Predicted positive does not match actual negative

False Negative (FN) (“Type II Error”): Predicted negative does not match actual positive

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218

		Predicted Label			
		A	B	C	D
Actual Label	A	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

True Positive (TP): Predicted positive matches actual positive

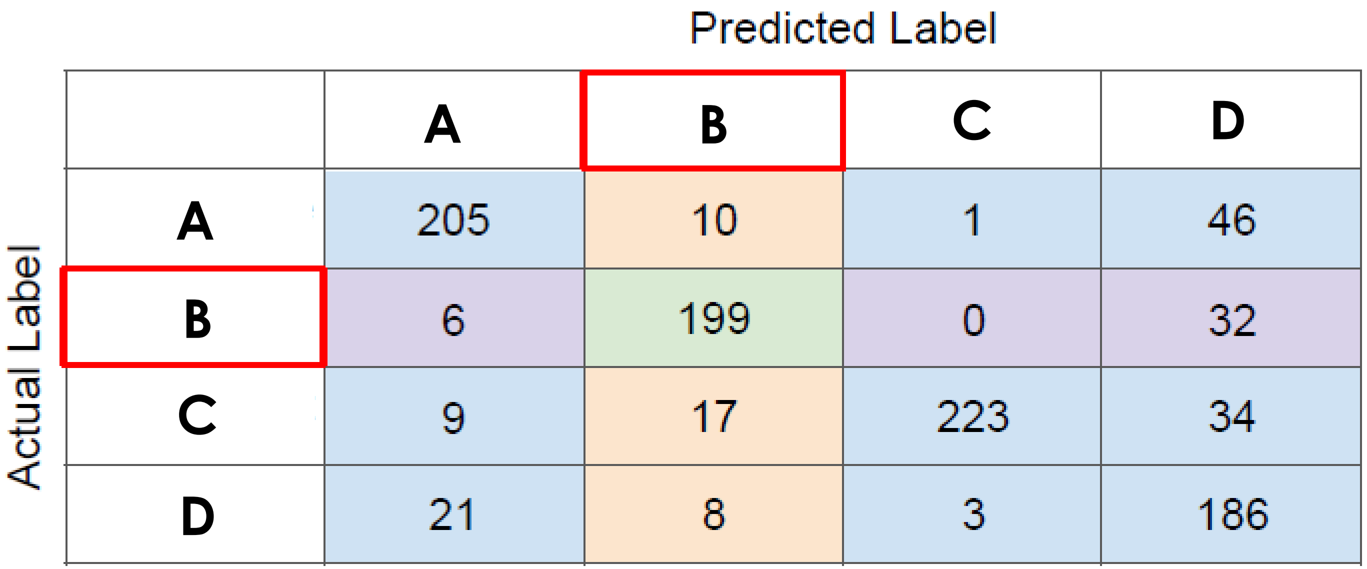
True Negative (TN): Predicted negative matches actual negative

False Positive (FP) (“Type I Error”): Predicted positive does not match actual negative

False Negative (FN) (“Type II Error”): Predicted negative does not match actual positive

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



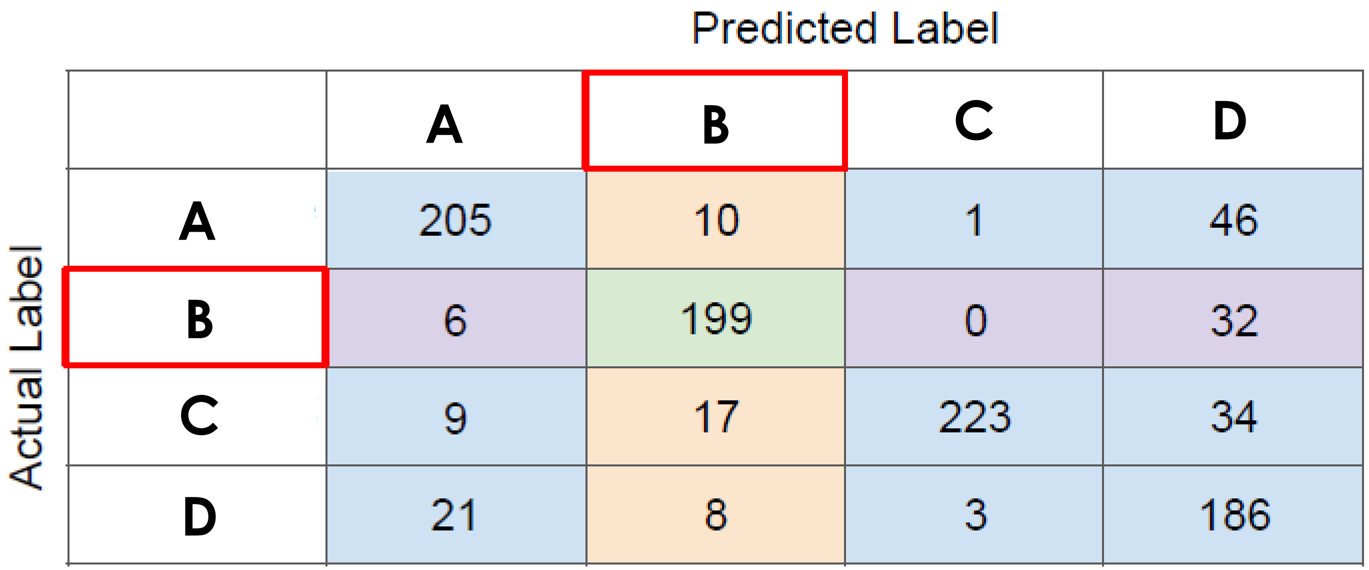
		Predicted Label			
		A	B	C	D
Actual Label	A	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

Per class Accuracy:

$$\boldsymbol{Accuracy_B} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{199+728}{1000} = 0.927$$

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



Predicted Label

Actual Label

	A	B	C	D
A	205	10	1	46
B	6	199	0	32
C	9	17	223	34
D	21	8	3	186

True Positive Rate (Sensitivity, Recall, Hit Rate):

$$\boldsymbol{TPR_B} = \frac{TP}{TP + FN} = \frac{199}{199 + 38} = 0.840$$

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218

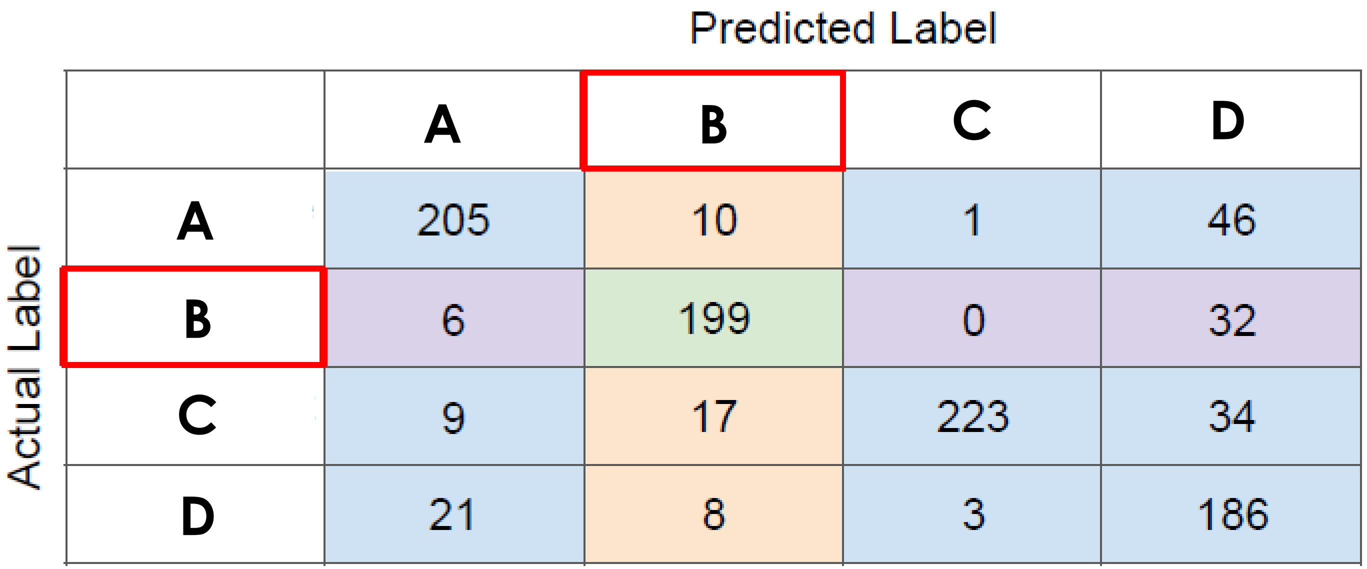
		Predicted Label			
		A	B	C	D
Actual Label	A	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

True Negative Rate (Specificity, Selectivity):

$$\boldsymbol{TNR_B} = \frac{TN}{TN + FP} = \frac{728}{728 + 35} = 0.954$$

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



		Predicted Label			
		A	B	C	D
Actual Label	A	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

Positive Predictive Value (Precision):

$$\boldsymbol{PPV_B} = \frac{TP}{TP + FP} = \frac{199}{199 + 35} = 0.850$$

Confusion matrix

- Total number of examples: 1000
- Class A: 262   Class B: 237   Class C: 283   Class D: 218

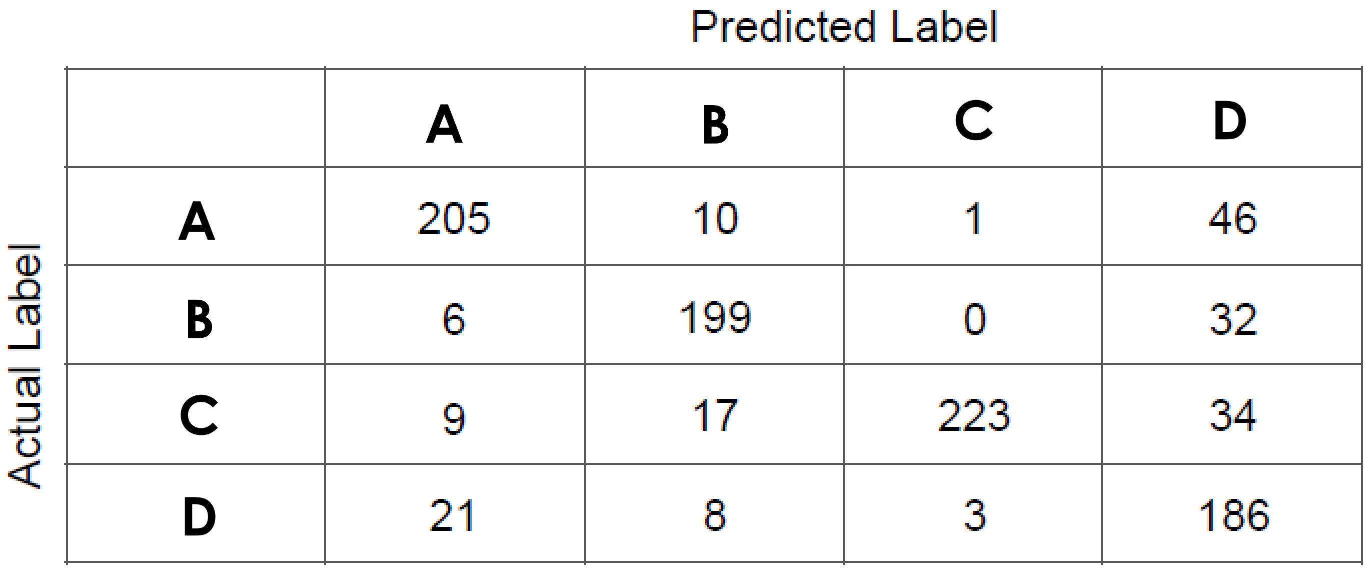
		Predicted Label			
		A	**B**	**C**	**D**
Actual Label	**A**	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

F1 score:

$$\boldsymbol{F1_B} = 2 \times \frac{PPV \times TPR}{PPV + TPR} = 2 \times \frac{0.850 \times 0.840}{0.850 + 0.840} = 0.845 = \frac{2 \times TP}{2 \times TP + FP + FN} = \frac{2 \times 199}{2 \times 199 + 35 + 38} = 0.845$$

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



Predicted Label

Actual Label

	A	B	C	D
A	205	10	1	46
B	6	199	0	32
C	9	17	223	34
D	21	8	3	186

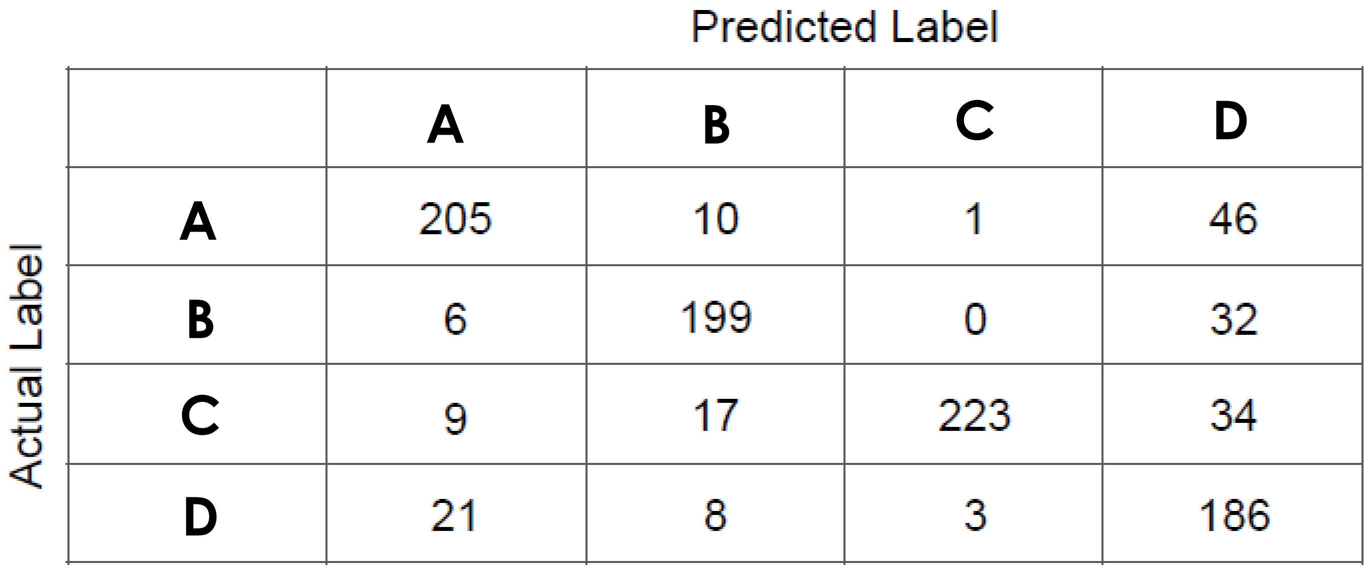
$$\boldsymbol{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\boldsymbol{F1\ Score} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

- Per-Class Accuracy:
- Per-Class F1 Score:
- Total Accuracy:
- F1 Score Average:

Confusion matrix

- Total number of examples: 1000
- Class A: 262 Class B: 237 Class C: 283 Class D: 218



		Predicted Label			
		A	**B**	**C**	**D**
Actual Label	**A**	205	10	1	46
	B	6	199	0	32
	C	9	17	223	34
	D	21	8	3	186

- Per-Class Accuracy: 0.907 0.927 0.936 0.856
- Per-Class F1 Score: 0.815 0.845 0.875 0.721
- Total Accuracy: 0.813
- F1 Score Average: 0.818

Text generation

Evaluation

ROUGE

Recall-Oriented Understudy for Gisting Evaluation

ROUGE calculates the intersection of the common n-grams between the auto-generated text (candidate) and the human generated text (reference):

- ROUGE-N
- ROUGE-L

Example:

Reference: التمر هو ثمرة أشجار النخيل

Candidate: أشجار النخيل تثمر التمر

ROUGE

Recall-Oriented Understudy for Gisting Evaluation

ROUGE-N: measures the number of matching n-grams between the candidate and the reference

$$Recall = \frac{Number\ of\ common\ n - grams\ between\ candidate\ and\ reference}{Total\ number\ of\ n - grams\ in\ reference}$$

$$Precision = \frac{Number\ of\ common\ n - grams\ between\ candidate\ and\ reference}{Total\ number\ of\ n - grams\ in\ candidate}$$

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

ROUGE-N	Reference	Candidate	Recall	Precision
ROUGE-1	التمر – هو – ثمرة – أشجار - النخيل	أشجار – النخيل – تثمر - التمر	3/5	3/4
ROUGE-2	التمر هو – هو ثمرة – ثمرة أشجار – أشجار النخيل	أشجار النخيل – النخيل تثمر – تثمر التمر	1/4	1/3

ROUGE

Recall-Oriented Understudy for Gisting Evaluation

ROUGE-L: measures the longest common subsequence (LCS) of words (not necessarily consecutive) between the candidate and the reference

$$Recall = \frac{Length\ of\ LCS}{Total\ number\ of\ 1-grams\ in\ reference}$$

$$Precision = \frac{Length\ of\ LCS}{Total\ number\ of\ 1-grams\ in\ candidate}$$

ROUGE-L	Reference	Candidate	Recall	Precision
	التمر هو ثمرة **أشجار النخيل**	**أشجار النخيل** تثمر التمر	2/5	2/4