Guidebook to Exercise 1

Ivan Habernal

2025-10-15

1 True and Predicted Labels

True Positive (TP)

Definition: The number of instances where the model correctly predicts the *positive* class.
Example: The model predicts "verb" and the true label is also "verb".
Interpretation: Correct positive prediction.

True Negative (TN)

Definition: The number of instances where the model correctly predicts the *negative* class.
Example: The model predicts "not verb" and the true label is indeed "not verb".
Interpretation: Correct negative prediction.

False Positive (FP)

Definition: The number of instances where the model incorrectly predicts the *positive* class when it should have predicted *negative*.
Example: The model predicts "verb", but the true label is "noun".
Interpretation: Incorrectly claiming something is positive.

False Negative (FN)

Definition: The number of instances where the model fails to predict the *positive* class when it should have.
Example: The model predicts "noun", but the true label is "verb".
Interpretation: Missed positive instance.

Term	Meaning	Model Prediction	Actual Class	Interpretation
TP	True Positive	Positive	Positive	Correct positive prediction
TN	True Negative	Negative	Negative	Correct negative prediction
FP	False Positive	Positive	Negative	Incorrectly predicted as positive
FN	False Negative	Negative	Positive	Missed a positive instance

Table 1: Summary of TP, TN, FP, and FN.

2 Evaluation Metrics

Accuracy

Accuracy measures the overall correctness of the classifier.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

It can be misleading in imbalanced datasets, as a model that always predicts the majority class may still achieve high accuracy.

Precision

Precision measures how many of the predicted positive instances are actually positive.

$$P_i = \frac{TP_i}{TP_i + FP_i}$$

A high precision means the model is cautious and rarely makes false positive predictions.

Recall

Recall measures how many of the actual positive instances were correctly predicted.

$$R_i = \frac{TP_i}{TP_i + FN_i}$$

A high recall means the model is comprehensive and rarely misses true positives.

F1 Score

The F1 score combines precision and recall into a single metric:

$$F1_i = \frac{2 \times P_i \times R_i}{P_i + R_i}$$

It is useful when a balance between precision and recall is desired.

3 Micro, Macro, and Weighted Averages

When dealing with multiple classes, metrics can be aggregated in different ways:

Type	Description	Calculation
Micro	Aggregates contributions of all classes before computing metrics.	Compute global TP, FP, FN, then calculate precision, recall, and F1.
Macro	Treats all classes equally, regardless of their frequency.	Compute metrics per class, then take the average.
Weighted	Accounts for class imbalance by weighting each class's contribution.	Weighted average based on the number of true instances per class.

Table 2: Averaging methods for multi-class metrics.

Two approaches to macro F1 are often used:

1. Compute F1 for each class and average the results.
2. Compute the mean of precision and recall first, then calculate F1.

4 Confusion Matrix

A **confusion matrix** summarizes the performance of a classification model by comparing predicted labels with true labels. Each row represents the instances of an actual class, while each column represents the instances predicted by the model.

True \Predicted	NN	VB	ADJ
NN	True Positive (NN→NN)	Misclassified as VB	Misclassified as ADJ
VB	Misclassified as NN	True Positive (VB→VB)	Misclassified as ADJ
ADJ	Misclassified as NN	Misclassified as VB	True Positive (ADJ→ADJ)

Table 3: Example of a confusion matrix for POS tagging.

Diagonal entries represent correct classifications (true positives), while off-diagonal entries indicate misclassifications.

Implementation Guidance

When building a confusion matrix class in Python:

1. Represent the matrix as a 2D array (e.g., a NumPy array).
2. For each `(true_label, predicted_label)` pair, increment the corresponding cell.
3. Include functions to compute metrics such as precision, recall, and F1 for each class.

Basic Python concepts that may be useful:

- Lists, dictionaries, and NumPy arrays (can be done without NumPy)
- Loops (`for`, `enumerate`)
- String formatting for displaying tabular output

5 Part-of-Speech (POS) Tagging

Part-of-Speech tagging assigns a grammatical category to each word in a sentence. Example tags:

- **NN** → Noun (e.g., *dog, computer*)
- **VB** → Verb (e.g., *run, jump*)
- **ADJ** → Adjective (e.g., *blue, heavy*)

In this exercise, POS tagging is treated as a **multi-class classification** task, where each token belongs to one of the defined categories.

6 BLEU Metric

BLEU (Bilingual Evaluation Understudy) evaluates the quality of machine-translated text by comparing it to one or more reference translations.

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Where:

- p_n: n-gram precision
- w_n: weight for each n-gram (often equal)
- BP: brevity penalty (penalizes outputs that are too short)

Advantages:

- Widely used and standardized
- Efficient to compute
- Works well for large-scale automatic evaluation

Limitations:

- Sensitive to exact word overlap (ignores meaning)
- Penalizes valid paraphrases
- Not suitable for very short or creative outputs

For experimentation, see the Hugging Face BLEU metric demo.

7 ROUGE Metric

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures the overlap between a system-generated text and reference texts, primarily focusing on recall. It is commonly used for automatic summarization.

Common variants include:

- **ROUGE-1**: overlap of single words (unigrams)
- **ROUGE-2**: overlap of word pairs (bigrams)
- **ROUGE-L**: longest common subsequence between candidate and reference

Advantages:

- Well-suited for summarization tasks
- Focuses on content coverage (recall)

Limitations:

- Based on surface similarity
- Does not evaluate fluency or coherence
- May undervalue concise but accurate summaries

Explore it using the Hugging Face ROUGE metric demo.