

# Assignment 3. Product Category/ESCI Classification

DATA304: Big Data Analysis

**Due Date:** November 10, 2025

## General Instructions

The programming assignments in this course are designed to foster your ability to implement and apply the concepts introduced during class. All datasets and resources for the assignments are provided at the following link: [Link](#). You do not need to download new data for each assignment. Instead, simply place the newly provided Jupyter notebook files for each assignment into your existing `assignment_release` directory (or into the renamed directory, if you have changed it).

- **How to submit:**

- You need to submit your code via LMS for all assignments. For each assignment, submit **All** the released `.ipynb` files, including all modifications you have made.
- Assignments #1, #2, and #3 also require Kaggle submissions.
- Assignments #4 and #5 do not require Kaggle submissions.

Make sure you submit both LMS and Kaggle on time to receive the credit. All submissions are due by midnight (23:59 KST) on the deadline day.

- **How the assignments are graded:**

- For Assignments #1, #2, and #3, you will receive full credit as long as you outperform our baseline code.
- For Assignments #4 and #5, you will receive full credit as long as you submit your code on time.

This policy is intended to encourage you to solve the problem by yourself and to compare your results with others. Use the Q&A board on LMS for all assignment-related questions.

## Assignment Instruction

This assignment has three notebooks: `P3a.train-with-BERT-embedding-no-label.ipynb`, `P3b.ESCI-train-with-BERT-embedding.ipynb`, and `P3c.ESCI-fine-tuning-BERT (reference).ipynb`.

## P3a: Train with BERT embedding (No label)

We consider a realistic but challenging scenario: *what if we have no labeled data at all?* In many real-world applications, collecting labeled data is expensive and slow. Here, we explore how to bootstrap a classification system without *any* human-provided labels. Your task is to fill in the blanks and design solutions for this “zero-label” setting.

### Constructing Silver Labels

Without ground-truth labels, the first step is to generate silver labels that serve as approximate supervisions. You are encouraged to explore various silver labeling strategies, such as:

1. **Lexical similarity:** Compare product titles and descriptions with category names using lexical similarity (e.g., TF-IDF-based).
2. **Embedding similarity:** Compare BERT embeddings of products and category labels to measure semantic closeness.
3. **Ensemble approaches:** Combine multiple weak labeling signals (e.g., weighted voting between lexical-based and embedding-based similarity) to obtain more reliable silver labels.

### Learning with Silver Labels

Once silver labels are constructed, you may train a classifier as if these labels were real. To enhance robustness and generalization in this noisy setting, you may apply advanced learning strategies discussed in class, including but not limited to:

1. **Self-training:** Train an initial model using silver labels, then iteratively refine it by generating pseudo-labels for unlabeled data with high prediction confidence.
2. **Label embedding models:** Incorporate the semantic meaning of label names into the model by using label embeddings (e.g., inner-product classifiers).
3. **Consistency regularization:** Encourage the model to produce stable predictions under input perturbations (e.g., dropout, data augmentation). This technique helps prevent overfitting to noisy labels and promotes smoother decision boundaries.
4. **Stabilizing model predictions using Ensemble methods:** To mitigate noise from weak or unstable supervision, apply ensemble-based stabilization techniques. This can include *independent model ensembles* (training multiple models and aggregating predictions) or *temporal ensembles* (averaging predictions over time using exponential moving averages).

█ **Your Task:** Your task is to generate silver labels, then train a classifier with these labels. Submit your best results to Kaggle.

*Note:* Do **NOT** use the labeled training set provided in the previous notebook. In this assignment, you must assume that **no labeled data exists**.

## P3b: ESCI classification with BERT embedding

In this notebook, we will explore classification tasks on the **ESCI dataset** using precomputed BERT embeddings for queries and documents. We will build a lightweight model on top of these fixed embeddings, training a simple classifier on concatenated query and document embeddings to predict one of four ESCI labels:

E (Exact)   S (Substitute)   C (Complement)   I (Irrelevant)

Unlike previous notebooks where labels were attached only to documents (products), here labels are defined for each **query–document pair**. Thus, the same product may have different labels depending on the query.

### Example:

- **Query:** “iPhone 13 case”
  - “Apple iPhone 13 silicone case” → E (**Exact**)
  - “Apple iPhone 12 case” → S (**Substitute**)
  - “Apple AirPods” → C (**Complement**)
  - “Samsung Galaxy S21 case” → I (**Irrelevant**)

This setup is both more **challenging** and **realistic**, since relevance now depends on the *interaction* between the query and the product.

1. **[Part A] Standard Training:** Train a multi-layer classifier on concatenated query and document embeddings to predict the four ESCI labels (E, S, C, I).
2. **[Part B] Multi-Task Learning (MTL):** Extend the model with an auxiliary task to improve generalization. The main task predicts full ESCI labels, while the auxiliary task distinguishes between simplified relevance levels (e.g., *E vs SCI*, *ESC vs I*, or other variants) to guide better shared representations.

 **Your Task:** Carefully read and understand the provided code. Examine intermediate values by printing them out, and make sure you clearly understand what operations are being performed at each step and how they fit together in the overall process. No submission required.

## P3c: Fine-tuning BERT

In this notebook, we shift from representation-based models to **cross-encoders**, where a query and a document are jointly processed through the same BERT model. The key idea is that the model can attend to both query and document tokens together. Instead of computing separate embeddings, the cross-encoder takes the concatenated input:

[CLS] query tokens [SEP] document tokens [SEP]

and uses the [CLS] token representation for classification (e.g., predicting document relevance).

This approach is well-suited for ESCI (Exact, Substitute, Complement, Irrelevant) classification, as it captures fine-grained semantic relations (e.g., synonyms, negations, attributes) that embedding-based models often miss. While cross-encoders are more computationally expensive, they typically achieve higher accuracy for nuanced matching tasks.

**Note:** This notebook may take a long time to run since it repeatedly encodes texts with BERT. It is provided mainly for reference, so you are encouraged to review the workflow rather than execute every cell.

# Submission Guidelines

## Kaggle Submission

- Submit your prediction file in `.csv` format to Kaggle: [Link](#)
- Ensure that the filename of your submission follows the format: `studentID_assignment3.csv`
- Set your **Kaggle team name** to your AWS account ID (e.g., `korea-dt-xx`). **This is mandatory in order to receive credit in our grading system. Please be careful!**
- There is **no limitation on the number of Kaggle submissions.**

## LMS Submission

- Submit your source code (all `.ipynb` files) to the LMS.
- Make sure the submitted code can be executed without errors.