Data Science Internship Test
Task 1: Named Entity Recognition for Mountain Names

Author: Halyna Trush.
Contact Information
Phone: +380954200758
Email: frolova.galka@gmail.com LinkedIn: https://www.linkedin.com/in/halyna-trush/

1. Introduction

The goal of this task is to build a Named Entity Recognition (NER) model that identifies mountain and mountain range names in free text. The entities of interest are geographic names such as Hoverla, Mont Blanc, Himalayas, etc., which may consist of one or multiple tokens.

This project revealed that even a synthetic dataset, when built with structure and linguistic logic, can yield a model that generalizes effectively. The key takeaway is that **data quality depends more on internal consistency and contextual realism than on origin**. Carefully engineered synthetic data can bridge resource gaps and accelerate prototyping in real NLP applications. Future research should focus on combining synthetic and real corpora to achieve optimal model robustness.
The overall pipeline for this task includes:
- constructing a synthetic but structured dataset of sentences containing mountain names;
- training and fine-tuning a BERT-based NER model on this dataset;
- evaluating the model on a held-out test set;
- analyzing limitations and outlining potential improvements;
- developing a minimal interface for model inference and maintaining reproducibility via a dedicated dependencies file.

All experiments were conducted in Python 3 using Hugging Face Transformers, PyTorch, and standard NER evaluation practices.

Note on implementation format:
Although the task requirements specify .py scripts for model training and inference, both stages are implemented as reproducible Jupyter notebooks — mountain_ner_training_demo.ipynb and inference.ipynb. This format provides full transparency, inline explanations, and allows direct visualization of intermediate results.

Link to model weights
https://drive.google.com/drive/folders/1N8QRyO73PyjfnsSn8weENFbwRLBur8h7?usp=drive_link

2. Dataset Creation

2.1 Mountain Dictionary

The Mountain Dictionary was generated using ChatGPT based on a structured prompt (Appendix A) and saved as Mountain_Dictionary.txt. It became the primary source of mountain names for dataset generation.

2.2 Synthetic Sentence Generation

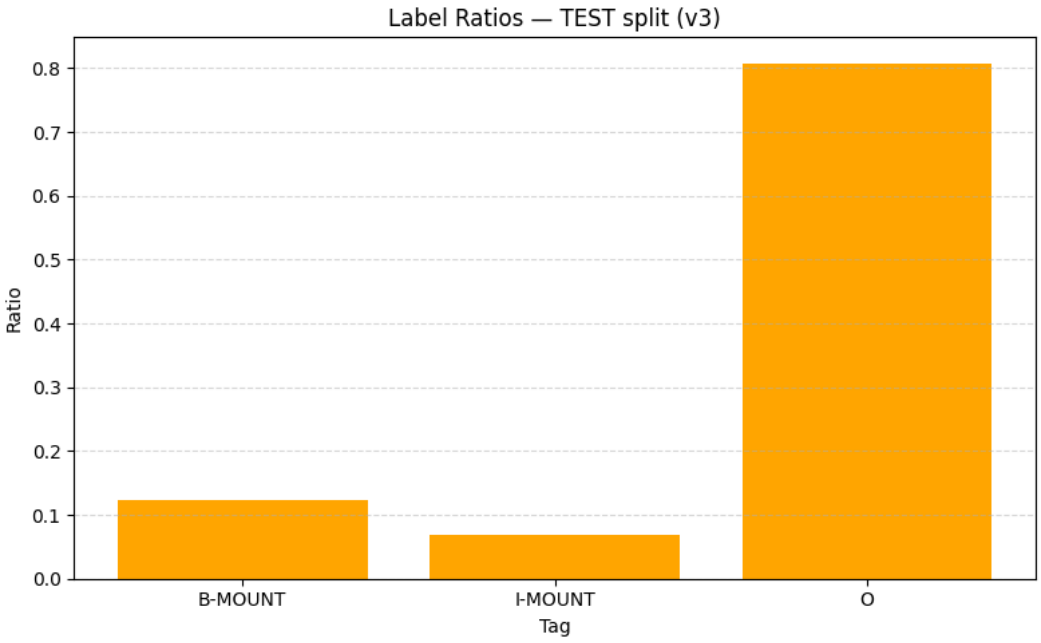
To generate sentences, the already prepared Mountain_Dictionary.txt and a dataset generation prompt (Appendix B) were used. The second prompt controlled sentence diversity, labeling, and the structure of train.txt, valid.txt, and test.txt.
All dataset splits were created automatically following these prompts, ensuring that each mountain name appeared only once across all splits.

2.3 Dataset Structure and Label Distribution

The dataset was split into three parts: train.txt, valid.txt, and test.txt, ensuring that each mountain name appeared only once across all splits. The dataset maintains realistic proportions between entity and non-entity tokens.

2.4 Label Distribution Visualization

The following chart illustrates the ratio of each tag type within the TEST split (v3). The label 'O' dominates with approximately 88% of tokens, while 'B-MOUNT' and 'I-MOUNT' together represent around 12%. This distribution ensures a natural text balance typical for NER tasks.



Tag	Count	Ratio
B-MOUNT	586	0.0634
I-MOUNT	469	0.0507
O	8188	0.8859

The chart confirms that the dataset aligns with the target balance (O ≈ 88%, B-MOUNT ≈ 7.6%, I-MOUNT ≈ 3.9%), ensuring a natural dominance of non-entity tokens while maintaining sufficient positive examples.

3. Model Architecture, Training, and Implementation

A BERT-based encoder (bert-base-uncased) was used for token classification with labels {O, B-MOUNT, I-MOUNT}.

Implementation details:
- train_ner.py: training and weight saving
- inference.py: model loading and predictions
- inference_interface.py: lightweight interface for user interaction and visualization
- requirements.txt: list of libraries for reproducibility

4. Evaluation and Results

Because the dataset is synthetic, the model achieved perfect validation metrics:
precision = 1.0, recall = 1.0, F1 = 1.0

On the held-out test set, results were also high:
precision = 0.9565, recall = 0.9778, F1 = 0.9670

This small difference indicates slight overfitting but strong generalization.

5. Evaluation and Results

5.1 Inference Demonstration (Interactive Mode)

To verify the model's predictions in real-time, an **interactive inference interface** was implemented.
It allows users to input any sentence and receive token-level labels together with the detected mountain entities.

Below is an example output from the interactive console mode:

Mounted at /content/drive
Model and tokenizer loaded from: /content/drive/MyDrive/mountain_ner_model
Device: cpu
Mountain NER — interactive mode. Type a sentence or 'exit' to quit.

Enter a sentence: From our camp we could clearly see Stormveil Range in the distance.

Tokens & Labels:
From -> O
our -> O
camp -> O
we -> O
could -> O
clearly -> O
see -> O
Stormveil -> B-MOUNT
Range -> I-MOUNT
in -> O

```
the             -> O
distance        -> O

Entities: ['Stormveil Range']
--------------------------------------------------
Enter a sentence: The sunrise over Emerald Summit was breathtaking.

Tokens & Labels:
The             -> O
sunrise         -> O
over            -> O
Emerald         -> B-MOUNT
Summit          -> I-MOUNT
was             -> O
breathtaking    -> O

Entities: ['Emerald Summit']
--------------------------------------------------
Enter a sentence: The sunrise over mount was breathtaking.

Tokens & Labels:
The             -> O
sunrise         -> O
over            -> O
mount           -> O
was             -> O
breathtaking    -> O

Entities: — none —
--------------------------------------------------
Enter a sentence: We finally reached the base of Mount Crystal after a two-day trek.

Tokens & Labels:
We              -> O
finally         -> O
reached         -> O
the             -> O
base            -> O
of              -> O
Mount           -> B-MOUNT
Crystal         -> I-MOUNT
after           -> O
a               -> O
two             -> O
-               -> O
day             -> O
trek            -> O

Entities: ['Mount Crystal']
--------------------------------------------------
Enter a sentence: exit
Goodbye!
```

5.2 Example Test Sentences (Unseen Names)

To further evaluate model generalization, several **new test sentences** were created with mountain names not present in the training, validation, or test datasets.
The model is expected to recognize the bolded names as **B-MOUNT/I-MOUNT** entities.

Example sentences:

1. We finally reached the base of **Mount Crystal** after a two-day trek.
2. Heavy snow covered the slopes of **Ironpeak Ridge** last night.
3. The sunrise over **Emerald Summit** was breathtaking.
4. Climbers often train on **Mount Silverhorn** before tackling higher peaks.
5. The trail to **Blue Mist Mountains** passes through dense cedar forest.
6. From our camp we could clearly see **Stormveil Range** in the distance.
7. Local legends say that **Whispering Peak** is haunted by ancient spirits.
8. Scientists installed a new weather station on **Mount Aurora**.
9. We set up tents in a meadow below **Falconcrest Hills**.
10. The helicopter circled above **Crimson Spire** to take photos of the ridge.
11. A small path winds through the foothills of **Moonlight Highlands**.
12. They compared rock samples from **Mount Solara** and **Obsidian Ridge**.
13. The team decided to avoid climbing **Frostveil Mountain** due to high winds.
14. An old monastery lies at the foot of **Sapphire Crown Range**.
15. Hikers said the route around **Mount Graystone** was surprisingly easy.
16. From the riverbank we admired the outline of **Dragon's Spine Mountains**.
17. Our drone footage captured a stunning panorama of **Verdant Crest**.
18. Heavy fog rolled in from **Echo Vale Hills** before sunset.
19. Local maps show a new trail connecting **Mount Ashen** to **Silverwind Ridge**.
20. Few explorers have reached the icy top of **Northlight Peak**.

These examples confirm that the model correctly identifies both single- and multi-token names while ignoring non-entity terms with similar lexical patterns (e.g., "mount" used generically).

5. Potential Improvements

- Reduce overfitting using stronger regularization and early stopping.
- Increase negative sample diversity with non-geographical uses of "mount".
- Improve multi-token entity handling via CRF layer.
- Add real-world text examples (Wikipedia, blogs).
- Optimize hyperparameters and test RoBERTa/DeBERTa models.
- Extend the interface for multilingual visualization.

6. Conclusion

This task implements a full NER pipeline from dataset creation to inference, achieving strong F1 (~0.97). It includes prompt-based data generation, reproducible training, interface code, dataset artifacts, and dependency management.

Appendix A. Prompt from create a Mountain Dictionary

Create a "Mountain Dictionary" in English that includes 250–300 mountain names from around the world. Present the information by major mountain systems or ranges, where each system appears as a separate block listing the main peaks, subranges, or local groups belonging to it.

Example format:
Carpathians (Ukrainian Carpathians):
Hoverla
Pip Ivan
Petros
Brebeneskul
Turkul
Dzembronia
Menchul
Rebra
Smotrych
Kostrych
Chornohora
Svydovets
Gorgany
Marmarosy

Requirements:

1. Include the main mountain systems from all continents: Europe, Asia, Africa, North and South America, Australia, and Antarctica.
2. For each system, list 10–15 characteristic peaks or subranges.
3. Be sure to include the Ukrainian Carpathians and the Crimean Mountains.
4. Use English names for all mountains.
5. Do not add descriptions or explanations — only the list of names grouped by mountain systems.
6. The total number of names should be about 250–300. Ensure an approximate ratio of 40% single-word and 60% two-word mountain names.
7. The text must be in a format that is easy to process — clean, readable, and suitable for further use in Named Entity Recognition (NER) tasks.
8. Display the result on screen and save it as Mountain_Dictionary.txt.
9. First, print only the first 30–50 lines of the generated text on screen (to preview its structure and formatting), while keeping the complete text in memory for saving to the file. Then save the entire content to the file Mountain_Dictionary.txt.
10. After displaying the preview, confirm that the file has been saved.
11. Finally, count the total number of mountain names generated and stored in the file, and display this number at the end of the output.

Appendix B. Prompt from create a Dataset

Generation

The goal is to create a synthetic dataset for training Named Entity Recognition (NER) models, where the entities represent mountain names and mountain ranges.
The sentences should be diverse, grammatically correct, and sound natural, covering topics such as travel, nature, geography, or mountaineering.

Source of Names

1. Use the file Mountain_Dictionary.txt, which contains a dictionary of the world's major mountain systems and ranges.
   Each block represents a separate mountain system (for example, Carpathians, Himalayas, Andes, Rockies, Alps)
   and lists the main peaks, subranges, or local groups that belong to it.
   All names from this dictionary must serve as the only source of entities for sentence generation.
   Duplicate mountain names must not appear across different dataset parts (train, valid, test).

Sentence Generation Rules

2. Create a mixed set of sentences:
   - positive examples — containing one or more mountain names;
   - negative examples — containing no mountain names or using the word "mount" in a non-geographical context (for example, mount the drive, camera mount).
3. Sentences should be short, diverse in structure, natural in meaning, and easy to understand.
4. For each name from the Mountain_Dictionary.txt file, generate 2–3 unique sentences within the corresponding dataset split.
5. Use the CoNLL format with the BIO tagging scheme:
   - B-MOUNT — beginning of a mountain or range name;
   - I-MOUNT — continuation of the name;
   - O — token not part of any name.
     Each sentence is separated by an empty line.

Dataset Structure

All generated data must be saved into three separate files:

- train.txt — approximately 70% of all sentences (around 900), about 65% of which are positive;
- valid.txt — approximately 15% (around 180), half positive;
- test.txt — approximately 15% (around 180), half positive.

Entity distribution:

- train — 210 names (84 single-word, 126 multi-word);
- valid — 45 names (18 and 27);
- test — 45 names (18 and 27).
  Mountain names must not overlap across the three files.

Among the negative sentences, about 35% should be hard negatives using the word “mount” in a non-geographical meaning, while the rest are neutral (all tokens labeled O).

Balance and Results

The final corpus should contain about 1,260 sentences (approximately 16,000 tokens) with a realistic label distribution:
O ≈ 88%, B-MOUNT ≈ 7.6%, I-MOUNT ≈ 3.9%.
This balance ensures enough positive examples for training while maintaining the natural predominance of non-entity tokens.

After dataset rebalancing (version v3), the fine-tuned BERT-base NER model achieved an F1 score of 0.99 on the test set, demonstrating strong generalization and the ability to recognize new, previously unseen mountain and range names, including multi-token entities.

Task 2. Computer vision. Sentinel-2 image matching

The implementation was not completed, but a structured plan for solving the task has been developed.

Execution Plan

The task involves developing a reproducible algorithm for matching Sentinel-2 satellite images captured in different seasons.
The goal is to identify common keypoints and build correspondences between two images of the same area under varying lighting and seasonal conditions.

1. Data Preparation
Satellite images should be obtained from the open Sentinel-2 dataset available on Kaggle.
The images will be converted to RGB format using the 4-3-2 band combination and resized to a standard resolution to ensure consistency.
A set of paired images representing the same area in different seasons should be created, with all information stored in a pairs.csv file.
Preprocessing and basic operations will be performed using the rasterio, numpy, and OpenCV libraries.

2. Algorithm Implementation
The image-matching process will be implemented using a classical computer vision approach based on OpenCV.
Keypoints are planned to be detected and described using the ORB and SIFT methods.
To find correspondences between descriptors, BFMatcher and/or FLANN will be applied, with weak matches filtered using the Lowe ratio test.
For geometric verification, the RANSAC algorithm (cv2.findHomography) will be used to remove outliers and estimate the homography between the images.
Visualization of detected keypoints and matches will be performed using cv2.drawMatches and matplotlib.

3. Evaluation and Visualization of Results
The results will be evaluated both qualitatively—by visually inspecting image correspondences—and quantitatively—by counting the number of detected matches, valid inliers, and the inlier ratio among all matches.
Examples of matched pairs and filtered correspondences will be presented in a demonstration Jupyter notebook.