Report on possible improvements in the results of both tasks.

Specific suggestions in the Summary sections

Task 1. NLP NER

Breaking down the task

Let's analyze the task.
- We need to train a Named Entity Recognition (NER) model to identify the names of mountains in texts.

From the above, we can identify a specific task - Named entity Recognition, as well as the entities that we need to identify: Mountain, O (mountain, not mountain).

Options

There are quite a few solutions for NER, they can be divided into pre-trained models and models that need to be trained from scratch.

just to name a few:
- Pre-trained: Hugging face bert, SpaCy, pytorch pretrained bert, etc.

libraries that allow you to train models from scratch:
- pytorch bert-pretrained, SpaCy, Hugging Face Transformers.

It is worth noting that these are not all possible options, there are many more, but these are the most popular ones.

Pre-Trained BERT

So, let's analyze these options.
Hugging Face has two types of pre-trained berts: small and large, as well as task-specific berts trained on different datasets.
The bert-base-NER model trained on CoNLL-2003 Named Entity Recognition would be suitable for us.
With the help of Fine-Tuning, we could train the model to recognize mountain names. There are two models for ner: small and large.
The pre-trained models are better at the task. Similarly, the larger model will perform better.
This option is good because the model already knows a lot of words and their context, so it doesn't need a large dataset for fine-tuning.

SpaCy, training from scratch

Until 2019, SpaCy allowed using its own pre-trained model bert uncased. It is similar to the previous version. But later, only empty models became available, which can be trained on your own tasks. The accuracy of the transformers provided by SpaCy may differ from bert by Hugging Face, and if we compare two different transformers, bert is likely to win in terms of accuracy.

GPT vs BERT

In addition to BERT, there are other transformers that are suitable for this task, such as GPT.
But it is important to understand the difference between them.
BERT uses only the encoder part of the Transformer architecture, processing the text in a bidirectional manner. GPT-1, on the other hand, uses only the decoder part of the Transformer architecture, processing the text in a unidirectional manner from left to right.
Simply put, BERT is designed for context detection, while gpt is designed to generate human-like text in response to a prompt.

BERT is an excellent tool for this task, as it is one of the tasks for which it was designed.

But it has alternatives in the same field, such as RoBERTa.

RoBERTa

The difference in a nutshell:
BERT uses two pretraining objectives - masked language modeling (MLM) and next sentence prediction (NSP). It randomly masks words in a sentence and trains the model to predict the masked words. NSP involves predicting whether two sentences follow each other in the training data.

RoBERTa removes the NSP objective and trains only on the MLM task. It also uses dynamic masking, meaning that different training runs use different masked tokens for the same input.

RoBERTa is trained on a much larger corpus of text and uses a larger batch size during training, which helps in capturing more diverse and nuanced language patterns.

RoBERTa trains on longer sequences without truncation, which helps in capturing dependencies over larger contextual windows.

Removing the NSP task allows RoBERTa to focus solely on masked language modeling, which is believed to be more beneficial for various NLP tasks.

Another option is XLNet.

XLNet

XLNet has a similar architecture to BERT. However, the major difference comes in it's approach to pre-training.

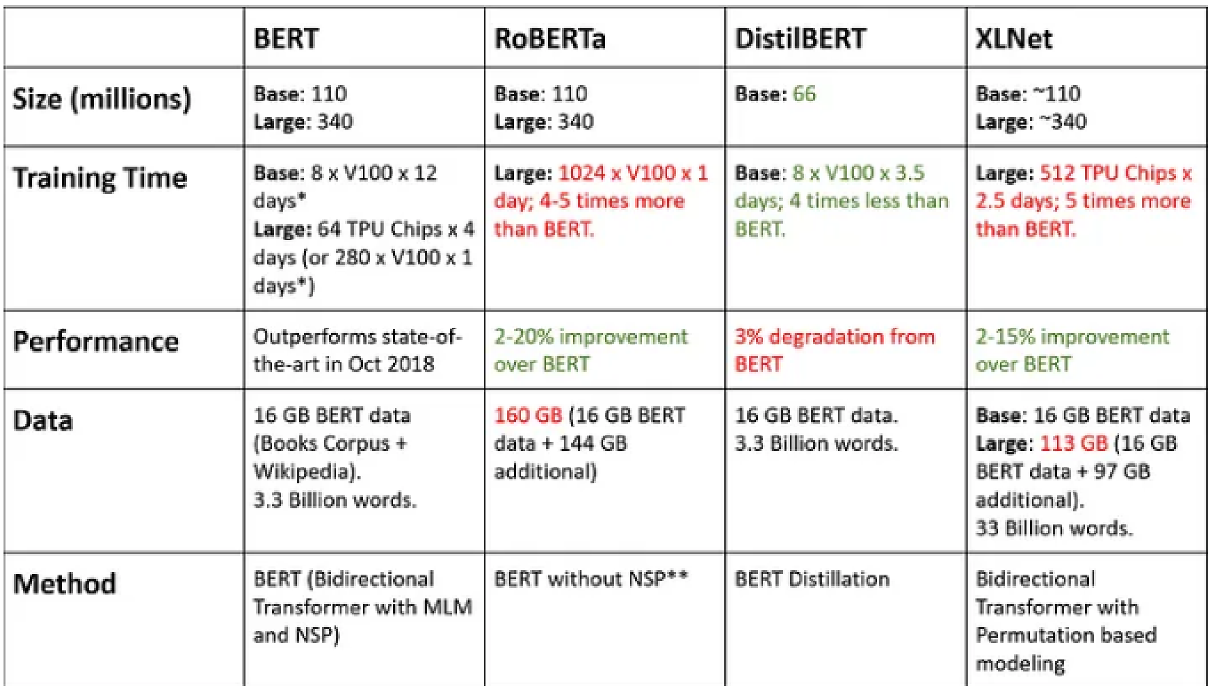
BERT is an Autoencoding (AE) based model, while XLNet is an Auto-Regressive (AR). This difference materializes in the MLM task, where randomly masked language tokens are to be predicted by the model. To better understand the difference, let's consider a concrete example [New, York, is, a, city].
Suppose both BERT and XLNet select the two tokens [New, York] as the prediction targets and maximize log(New, York | is, a, city). Also suppose that XLNet samples the factorization order [is, a, city, New, York]. In this case, BERT and XLNet respectively reduce to the following objective functions:

J{BERT} = log(New | is, a, city) + log(York | is, a, city) and

J{XLNet} = log(New | is, a, city) + log(York | New, is, a, city)

Notice that XLNet is able to capture the dependency between the pair (New, York), which is omitted by BERT. Although in this example, BERT learns some dependency pairs such as (New, city) and (York, city), it is obvious that XLNet always learns more dependency pairs given the same target and contains "denser" effective training signals.

Summary



	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base**: 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large**: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large**: 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large**: 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

Medium article about BERT, RoBERTa, XLNet, DistiliBERT differences

In summary, the NER task can be performed using a better architecture (RoBERTa or XLNet), which would improve accuracy. But it would also slow down the training.

If we talk not only about changing the architecture, the preparation of a high-quality dataset could significantly improve the recognition accuracy in our case.

Used literature:

1. Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Daniel Jurafsky, Stanford University
2. LinguisticsNeural Architectures for Named Entity Recognition
3. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
4. Named Entity Recognition (NER) Benchmarks
5. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition
6. Single-/Multi-Source Cross-Lingual NER via Teacher-Student Learning on Unlabeled Data in Target Language
7. Medium article about BERT, RoBERTa, XLNet, DistiliBERT differences
8. Unsupervised Cross-lingual Representation Learning at Scale

Task 2 Feature Matching

Breaking down the task

In this task, you will work on the algorithm (or model) for matching satellite images. For the
dataset creation, you can download Sentinel-2 images from the official source here or use our
dataset from Kaggle. Your algorithm should work with images from different seasons. For this

Options comparison

Harris Corner Detector identifies points where there is a significant change in intensity in two directions. Typically uses a small window around the corner to describe local appearance. It's focuses on corners in images and is sensitive to intensity changes.

SIFT is scale-invariant and provides robustness to changes in scale and orientation. It detects keypoints at multiple scales and orientations. It's utilizes histograms of gradient orientations in a local region around each keypoint.

SURF uses an integral image for fast feature detection at different scales. Describes local appearance using Haar wavelet responses. SURF is designed for efficiency and is faster than SIFT, making it suitable for real-time applications.
FAST identifies corners based on pixel intensity comparisons. Typically uses simple binary descriptors. FAST is focused on speed and efficiency, making it suitable for real-time applications.

BRIEF relies on FAST for interest point detection. Utilizes binary strings to describe local appearance. BRIEF is binary and efficient but may lack robustness in handling variations in scale and orientation.

ORB combines FAST for interest point detection and BRIEF for feature description. Adds rotation invariance to BRIEF descriptors.

ORB is designed to be fast and efficient with improved rotation invariance compared to BRIEF.

Image pre-processing

There are many ways to pre-process images to help feature extractors perform better.
Some of the most popular techniques are image resizing and grayscale conversion.

SURF can work with color images, and the use of grayscale simplifies processing and reduces the computational load.

Also, images may be noisy, for which you can use Gaussian blurring.
Histogram equalization helps to sharpen the image.

Much more interesting is the Region of Interest (ROI) selection,
ROI pooling is a key technique for addressing the challenge of fixed-size input requirements in object detection networks, allowing for the processing of proposals with varying shapes by converting them into fixed-size feature maps through max-pooling operations.

In the object detection process, the entire image is input into a Convolutional Neural Network (CNN) to detect Regions of Interest (RoI) on the feature maps. Each RoI is then isolated using an ROI pooling layer, and the resulting feature vectors are fed into fully-connected layers. These vectors are utilized by a softmax classifier for object detection and a linear regressor for modifying the coordinates of the bounding box.

Summary:

To improve the results of this task, you can use the SURF model for large amounts of data by pre-processing the data (grayscaling, resizing, ROI, etc.)

Literature

1. Harris Corner Detector
2. Introduction to SIFT(Scale Invariant Feature Transform)
3. Introduction to SURF (Speeded-Up Robust Features)
4. Introduction to FAST (Features from Accelerated Segment Test)
5. Introduction to BRIEF(Binary Robust Independent Elementary Features)
6. Introduction to ORB (Oriented FAST and Rotated BRIEF)
7. Image Pre-Processing for faster Image Recognition
8. Region of Interest Pooling