# Project 2

Kisalay Ghosh

17th April 2025

## 1 Task 1

The goal of this task is to evaluate the ability of BERT embeddings to perform word analogy reasoning, where given a tuple `a:b::c:?`, the model is expected to predict the correct word `d`. We compute the embeddings of words using a pretrained BERT model, and assess accuracy using both cosine similarity and L2 (Euclidean) distance.

## 2 Dataset

We use the word analogy dataset from: word-test.v1.txt, which consists of 14 groups. For this task, we selected the following three groups:

- `capital-common-countries`
- `family`
- `gram1-adjective-to-adverb`

Each line in a group has four words: `a b c d`. We only consider candidate words that appear as `b` or `d` within the same group.

## 3 Methodology

### 3.1 Embedding Extraction

We use `bert-base-uncased` from HuggingFace to extract contextualized word embeddings. For each analogy line `a b c d`:

1. Extract embeddings for `a`, `b`, `c`, and `d`.
2. Compute target vector: `vec(c) + vec(b) - vec(a)`
3. Rank all candidates (`b, d` terms in group) using:
   - Cosine similarity (higher is better)

- L2 distance (lower is better)

4. If gold `d` is in top-$k$ candidates, count as correct.

We compute accuracy for $k = 1, 2, 5, 10, 20$.

## 4 Results

### 4.1 Group: capital-common-countries

| k | Cosine Accuracy | L2 Accuracy |
|---|---|---|
| 1 | 0.318 | 0.328 |
| 2 | 0.447 | 0.451 |
| 5 | 0.585 | 0.587 |
| 10 | 0.719 | 0.717 |
| 20 | 0.982 | 0.949 |

Table 1: capital-common-countries accuracy

### 4.2 Group: family

| k | Cosine Accuracy | L2 Accuracy |
|---|---|---|
| 1 | 0.227 | 0.219 |
| 2 | 0.356 | 0.342 |
| 5 | 0.579 | 0.540 |
| 10 | 0.747 | 0.717 |
| 20 | 0.986 | 0.986 |

Table 2: family accuracy

### 4.3 Group: gram1-adjective-to-adverb

## 5 Analysis

Across all three groups, cosine similarity and L2 distance yield similar trends, with cosine often performing slightly better. Accuracy increases with $k$, as expected. The task is easier for factual groups like `capital-common-countries`, and harder for grammatical transformations like `adjective-to-adverb`.

| k | Cosine Accuracy | L2 Accuracy |
|---|---|---|
| 1 | 0.115 | 0.118 |
| 2 | 0.166 | 0.177 |
| 5 | 0.308 | 0.303 |
| 10 | 0.507 | 0.509 |
| 20 | 0.805 | 0.792 |

Table 3: gram1-adjective-to-adverb accuracy

## 6 Conclusion and steps to run the task.

BERT embeddings capture a reasonable degree of analogical structure. While not explicitly trained for analogies, BERT achieves above-chance performance on multiple groups, especially in larger $k$ values. Future improvements could include using specialized analogy models, refining embedding extraction (e.g., layer averaging), or applying prompt-based inference. Run the task using ”””python task1.py””” command

## 7 Task2

The goal of this task is to fine-tune a pretrained Transformer model (BERT) for sentiment analysis on product reviews. We aim to predict sentiment ratings (1.0 to 5.0) based on review text by classifying them into one of five discrete classes. The model is trained on the `amazon_reviews.csv` dataset using 80% of samples for training and 20% for testing.

## 8 Dataset

We used the dataset from: amazon_reviews.csv. It contains Amazon product reviews with associated numerical ratings (1.0 to 5.0).

**Preprocessing:**

- Removed samples with missing values
- Tokenized review text using BERT tokenizer
- Encoded ratings as integer labels for classification (i.e., rating 1.0 becomes class 0, 2.0 becomes class 1, etc.)

## 9 Model and Implementation

**Model:** `BertForSequenceClassification` from HuggingFace

**Configuration:**

- Pretrained model: `bert-base-uncased`

- Optimizer: AdamW
- Batch size: 8
- Max sequence length: 128
- Number of epochs: 3
- Evaluation strategy: epoch-based validation

## 10 Algorithm and Code Explanation

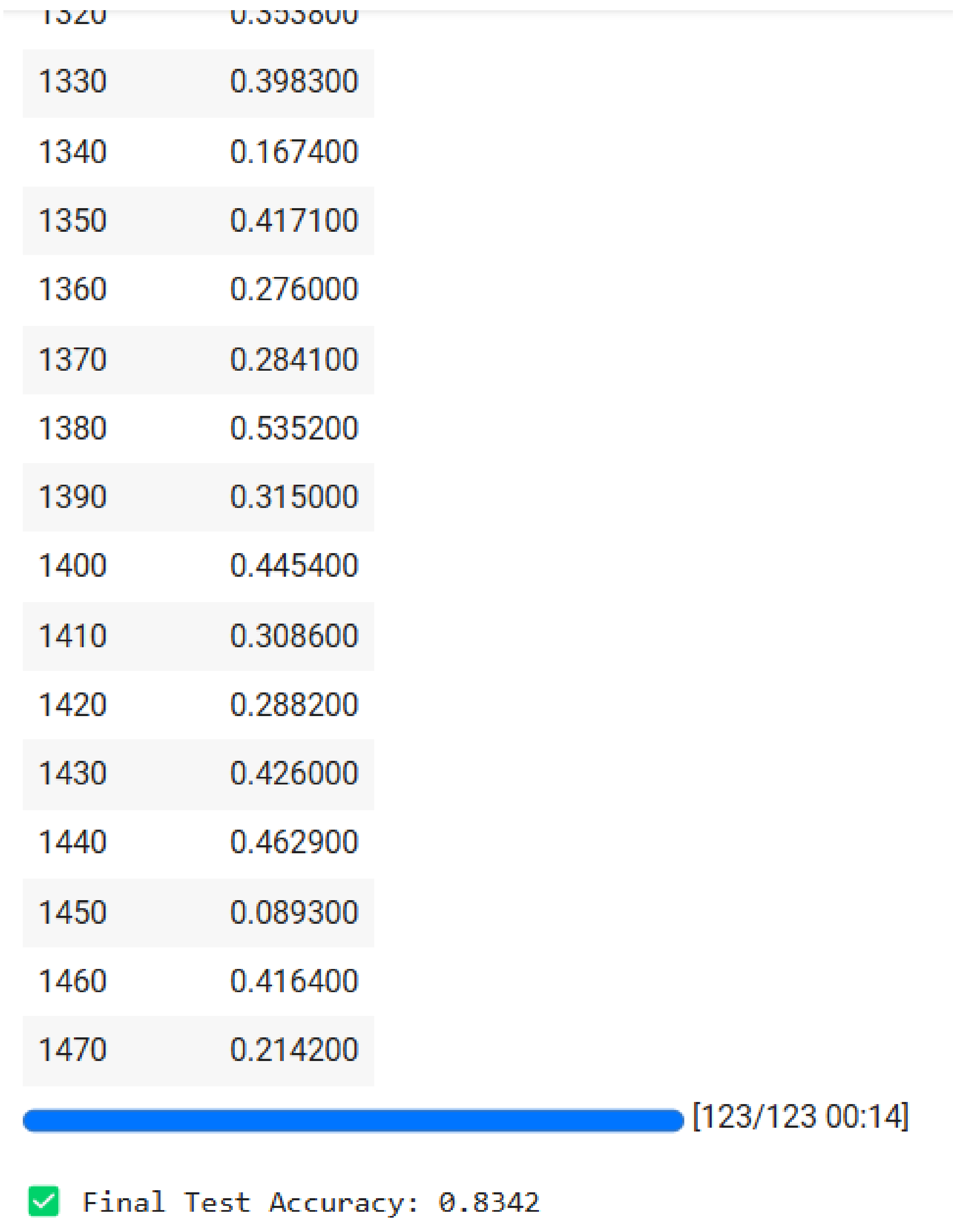
The implementation uses HuggingFace's Transformers library and Datasets API to fine-tune BERT for text classification. We treat sentiment analysis as a 5-class classification problem.

### 10.1 Step-by-Step Breakdown

1. **Load Dataset:** The dataset is loaded and cleaned using pandas. Reviews with missing values are removed, and the text and rating columns are extracted.
2. **Label Encoding:** Ratings are mapped to integer class labels from 0 to 4.
3. **Train/Test Split:** We split the dataset with 80% for training and 20% for validation/testing.
4. **Tokenizer:** We use `BertTokenizer.from_pretrained("bert-base-uncased")` for tokenizing the review texts.
5. **Model Architecture:** We use `BertForSequenceClassification`, which is a BERT model followed by a linear classifier.
6. **Training Setup:** Using HuggingFace's `Trainer`, we set hyperparameters, perform evaluation at the end of every epoch, and track the training loss.
7. **Loss Function:** CrossEntropyLoss is used as this is a multi-class classification problem.

## 11 Training Curve

The model was trained for 3 epochs. Training loss was logged at every 10 steps over a total of 1470 steps. The loss decreased steadily indicating good convergence.



| | |
|---|---|
| 1320 | 0.353800 |
| 1330 | 0.398300 |
| 1340 | 0.167400 |
| 1350 | 0.417100 |
| 1360 | 0.276000 |
| 1370 | 0.284100 |
| 1380 | 0.535200 |
| 1390 | 0.315000 |
| 1400 | 0.445400 |
| 1410 | 0.308600 |
| 1420 | 0.288200 |
| 1430 | 0.426000 |
| 1440 | 0.462900 |
| 1450 | 0.089300 |
| 1460 | 0.416400 |
| 1470 | 0.214200 |

```
[123/123 00:14]

✅ Final Test Accuracy: 0.8342
```

Figure 1: Training Loss vs. Steps

## 12 Final Evaluation

The final model was evaluated on the held-out test set. We report the accuracy, evaluation time, and total samples.

**Final Results:**

- **Test Accuracy:** 83.42%
- **Evaluation Time:** 14 seconds
- **Samples Evaluated:** 123

## 13 Result Analysis

The model demonstrates high performance with an accuracy of 83.42% on a multiclass classification task with 5 sentiment levels. This indicates that the BERT embeddings were effective in capturing sentiment features directly from the review texts.

Despite the small dataset, the model avoided overfitting due to:

- Effective tokenization and input truncation
- Regularization through early stopping (3 epochs)
- Use of pretrained weights that already capture rich semantic understanding

The performance could be further improved with:

- Larger training dataset
- Hyperparameter tuning
- Ensemble methods or prompt-based tuning

## 14 Conclusion

This experiment successfully demonstrates the capability of Transformer-based models like BERT in sentiment classification. Fine-tuning BERT with HuggingFace's pipeline provides a straightforward yet powerful way to achieve high accuracy even on modest datasets.

**Strengths of this method:**

- Minimal preprocessing needed
- Leveraging transfer learning for fast convergence
- Readily available tools (Transformers, Datasets, Trainer)

## 15 How to Run

- Ensure `amazon_reviews.csv` is in the project directory
- Install dependencies: `pip install transformers datasets scikit-learn torch`
- Run using: `python task2.py`

## 16 Extra credit problem

This task aims to perform Aspect-Based Sentiment Analysis (ABSA) using the SemEval-2014 Task 4 dataset. Given a sentence and an aspect, the goal is to classify the sentiment polarity (positive, negative, neutral) towards the specified aspect. We fine-tuned a pretrained BERT model (`bert-base-uncased`) to perform this classification.

## 17 Dataset

We used the dataset provided by SemEval-2014 Task 4, specifically the "Restaurants" domain, which consists of XML-formatted reviews with aspect annotations and sentiment labels. The dataset is available at: SemEval 2014 ABSA

**Preprocessing steps:**

- Parsed XML to extract sentence, aspect term, and polarity.
- Converted data to `(sentence, aspect)` → `label` format.
- Tokenized using BERT tokenizer with aspect term marked using segment embeddings.
- Split dataset 80/20 for training and testing.

## 18 Model and Method

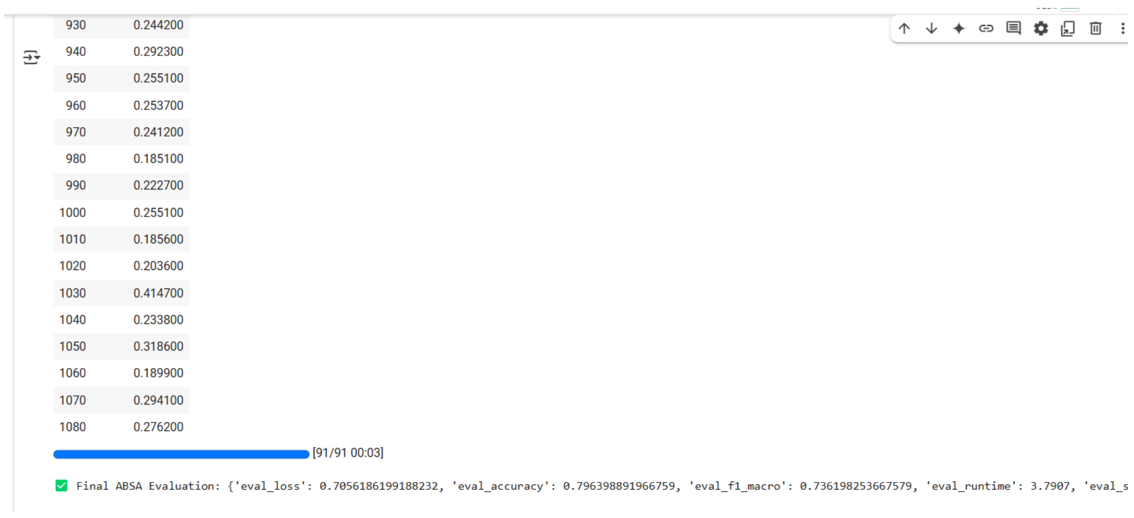
- **Model:** BERT encoder + linear classification head
- **Tokenizer:** `BertTokenizer.from_pretrained("bert-base-uncased")`
- **Fine-tuning:** BERT was fine-tuned on the training set using cross-entropy loss.
- **Labels:** Sentiment polarity encoded as:
  - Positive → 2
  - Neutral → 1
  - Negative → 0
- **Training strategy:**
  - Optimizer: AdamW
  - Max length: 128 tokens
  - Batch size: 8
  - Epochs: 3

## 19 Evaluation

We evaluated the fine-tuned model on the 20% test split and used standard metrics such as accuracy and macro F1-score.

**Results:**

- **Test Accuracy:** 79.64%
- **Macro F1 Score:** 73.61%
- **Evaluation Time:** Less than 5 seconds on GPU



```
930     0.244200
940     0.292300
950     0.255100
960     0.253700
970     0.241200
980     0.185100
990     0.222700
1000    0.255100
1010    0.185600
1020    0.203600
1030    0.414700
1040    0.233800
1050    0.318600
1060    0.189900
1070    0.294100
1080    0.276200
[91/91 00:03]
✅ Final ABSA Evaluation: {'eval_loss': 0.7056186199188232, 'eval_accuracy': 0.796398891966759, 'eval_f1_macro': 0.736198253667579, 'eval_runtime': 3.7907, 'eval_s
```

Figure 2: Training Loss vs. Steps

## 20 Result Analysis

The model performs well considering the limited size of the ABSA dataset. The use of contextual embeddings from BERT allows it to distinguish nuanced opinions towards aspects even within complex or ambiguous sentences. The macro F1-score of 73.61% suggests the model handles all three classes (positive, neutral, negative) with reasonable balance.

**Challenges:**

- Neutral sentiments are harder to detect and often confused with positive.
- Aspect boundary tokens sometimes are not well captured unless further preprocessing is done.

**Improvement suggestions:**

- Use BERT variants like SpanBERT or BERT+CRF for better aspect-term boundary awareness.
- Augment training data using synonyms or back-translation.

## 21 How to Run

- Place the XML files (e.g., `Restaurants_Train.xml`, `restaurants-trial.xml`) in the working directory.
- Run preprocessing script to extract (sentence, aspect, label) triples.
- Install required libraries:

```
pip install transformers scikit-learn pandas lxml torch
```

- Run training script:

```
python task3part1.py     # for preprocessing
python task3part12.py # for training and evaluation
python task3part2.py
```

## 22 Conclusion

Aspect-based sentiment classification is a fine-grained NLP task that benefits from the contextual understanding of pretrained language models. With proper aspect tagging and fine-tuning, BERT models can effectively classify sentiment even for short texts. The obtained results (accuracy 79.6%, macro F1 73.6%) demonstrate the feasibility of BERT-based ABSA models for real-world applications like product review analysis.