Central Asian University

Sentiment Classification of Uzbek E-commerce (Uzum Market) Customer Reviews

Main goal is to build and compare multiple **NLP Transformer** models
for sentiment classification of **Uzbek** reviews.

📌 Results of Transformer:

Transformer encoder model achieved a training **loss of 0.1891** on Uzbek review sentiment classification

Why Loss = 0.1891 Is Meaningful:

- Cross-entropy loss **close to 0** means:
- predicted probability mass concentrated on correct class
- For 3-class classification:
- random **guess ≈ 1.10 loss**
- Transformer **model ≪ random baseline**

Compared to a random classifier, the Transformer **significantly reduces uncertainty in predictions.**

📌 Why Uzbek-language based model

Motivation:

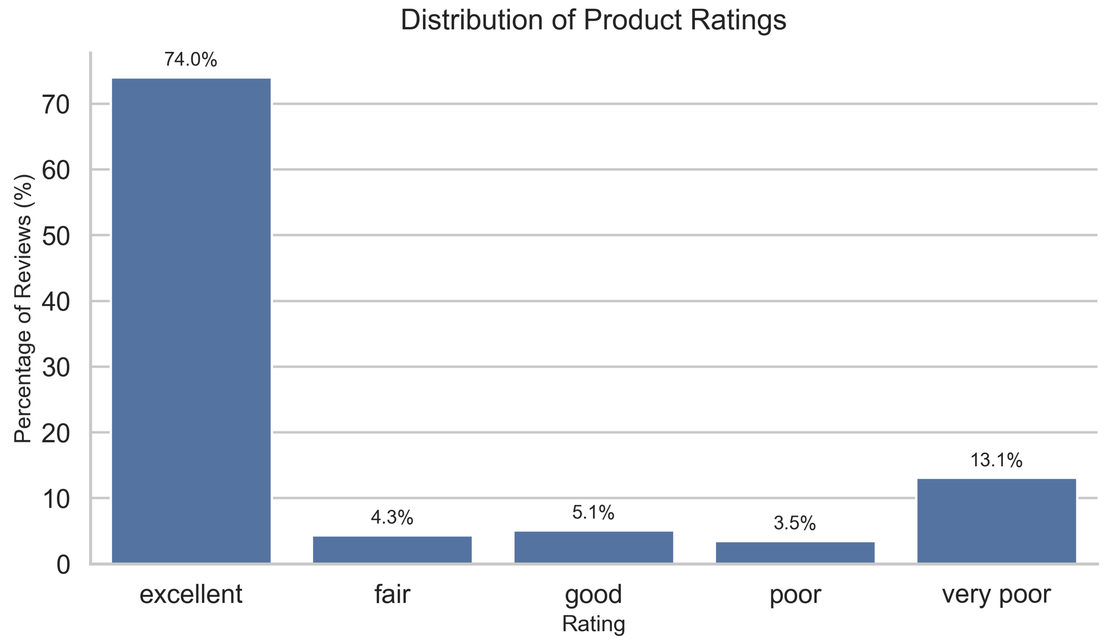
- Uzbek language is **low-resource** in NLP
- Real business value:
  - **automatic** customer satisfaction **monitoring**
  - **rating prediction**
  - feedback analysis for marketplaces (Uzum)

Challenges:

- noisy user-generated text
- mixed scripts (**Latin + Cyrillic)** and some **Russian mixed texts**
- non-standard characters

📌 Dataset Description:

- Source: **Uzum Market** Customer reviews
- Size: **~350k reviews**
- Class **imbalance**
- **Average** review length = **31 words**
- Language mixture (**Latin + Cyrillic+Russian**)



📌 Text Normalization:

This step significantly reduces **vocabulary size** and **noise**.

Why normalization is critical for Uzbek:

- different alphabets
- accented characters
- foreign symbols

What I did:

- Unicode-based filtering (\p{Latin}, \p{Cyrillic})
- Character-level normalization mapping
- Removal of unwanted symbols
- Preservation of spaces

```
Before: "Rahmat!!! Júdá yoqdi "
After:  "rahmat juda yoqdi"
```

📌 Baseline Models:

1. BoW + Linear Classifier

- CountVectorizer
- Linear layer
- Fast but ignores word order

2. Embedding + Mean Pooling

- Token embeddings
- Mean over sequence
- Simple semantic representation

3. Transformer Encoder Classifier

- Token embedding
- Positional embedding
- **4 Transformer blocks:**
  - Multi-head self-attention
  - Feed-forward network
  - Residual connections
  - LayerNorm
- Mean pooling over sequence
- Linear classifier head

📌 Model Complexity and Architecture:

Component	Trainable Parameters
Token Embedding Layer	1,920,000
Positional Embedding	5,120
Self-Attention Blocks	65,792
Feed-Forward Blocks	132,352
Classification Head	195
Total Parameters	**2,123,459**
Number of epochs	10
Learning rate	3×10^{-3}
Training time	~25 minutes on GPU

📌 Outputs from Model:

```
text = "oyimga ko'rsattim bo'larkan, yoqmasa kerak deb o'ylagandim"
probs = model.predict(text, tokenizer)[0].tolist()

out = np.argmax(probs)

print(probs)
print('excellent' if out == 2 else 'fair' if out == 1 else 'poor')
```

```
[0.0003321970871184021, 5.523517756955698e-05, 0.9996126294136047]
excellent
```

```
text = "oyimga ko'rsattim bo'larkan, yoqmasa kerak deb o'ylagandim, lekin o'zimga to'grisi vashe ishlashi yoqmadi"
probs = model.predict(text, tokenizer)[0].tolist()

out = np.argmax(probs)

print(probs)
print('excellent' if out == 2 else 'fair' if out == 1 else 'poor')
```

```
[0.9644067883491516, 0.0064508188515901566, 0.029142329469323158]
poor
```

Thank You

Bakhtiyor Bekmurodov - 230035

Source code: https://github.com/baxt1or/ml-final-exam/blob/main/scripts/train_transformer.py

LinkedIn: https://www.linkedin.com/in/bakhtiyor-bekmurodov-farhod-ogli/

Central Asian University