INLP_Project_Report

INLP Project Report

Project Overview

This project focuses on analyzing a Turkish electronics review dataset by translating it to English and applying various Natural Language Processing (NLP) techniques. The primary objectives include:

1. Data preprocessing and translation

2. Exploratory data analysis

3. Text representation using different embedding techniques:

Continuous Bag of Words (CBOW)

Skip-gram

TF-IDF (Term Frequency-Inverse Document Frequency)

Sbert

Data Preprocessing

Translation Process

The project begins with translating the Turkish electronics review dataset to English using Google Translator API. The implementation

in data_preprocessing.py handles this process:

The dataset is processed in batches to manage API limitations and prevent rate limiting

Long comments are split into chunks of 5000 characters to accommodate Google Translator's character limit

Both comments and their associated labels are translated from Turkish to English

The translated data is saved as translated_output_1.csv and translated_output_2.csv in the interim dataset directory

This translation step was crucial as it allowed us to apply English-based NLP techniques to the originally Turkish dataset, making it accessible for a wider range of tools and methods.

Text Preprocessing

Before generating embeddings, the text data undergoes several preprocessing steps:

Tokenization

: Breaking text into individual words using NLTK's word_tokenize function

Lowercasing

: Converting all text to lowercase to standardize the vocabulary

Stopword removal

: Eliminating common words (e.g., "the", "and", "is") that add little semantic meaning

Lemmatization

: Reducing words to their base form (e.g., "running" → "run") using WordNetLemmatizer

Special character removal

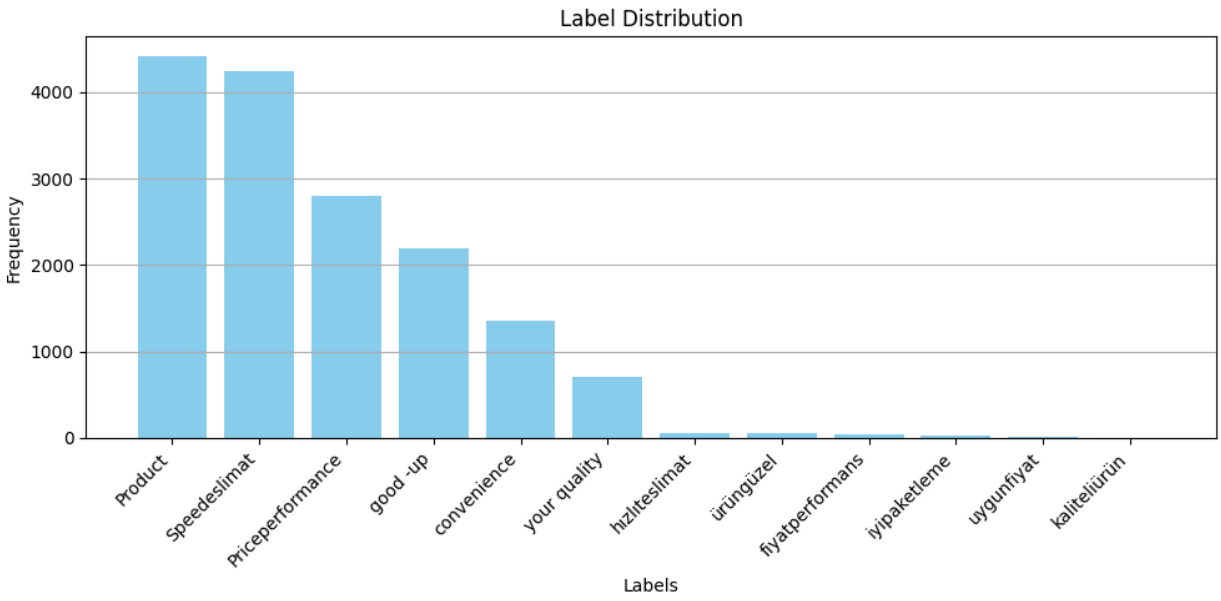
: Keeping only alphanumeric characters to reduce noise

The preprocessing steps significantly reduced vocabulary size while preserving semantic meaning, which is essential for creating meaningful embeddings.

Exploratory Data Analysis

The data_vis.py script performs exploratory data analysis on the translated dataset, generating visualizations to better understand the data characteristics:

Label Distribution Analysis



The label distribution analysis reveals the frequency of each label in the dataset:

Labels like "Product", "Priceperformance", and "Speedslimat" are heavily represented

Other labels like "iyipaketleme" and "kaliteliürün" appear much less frequently

This indicates significant class imbalance, which is critical to address in any classification task

This imbalance suggests the need for techniques such as:

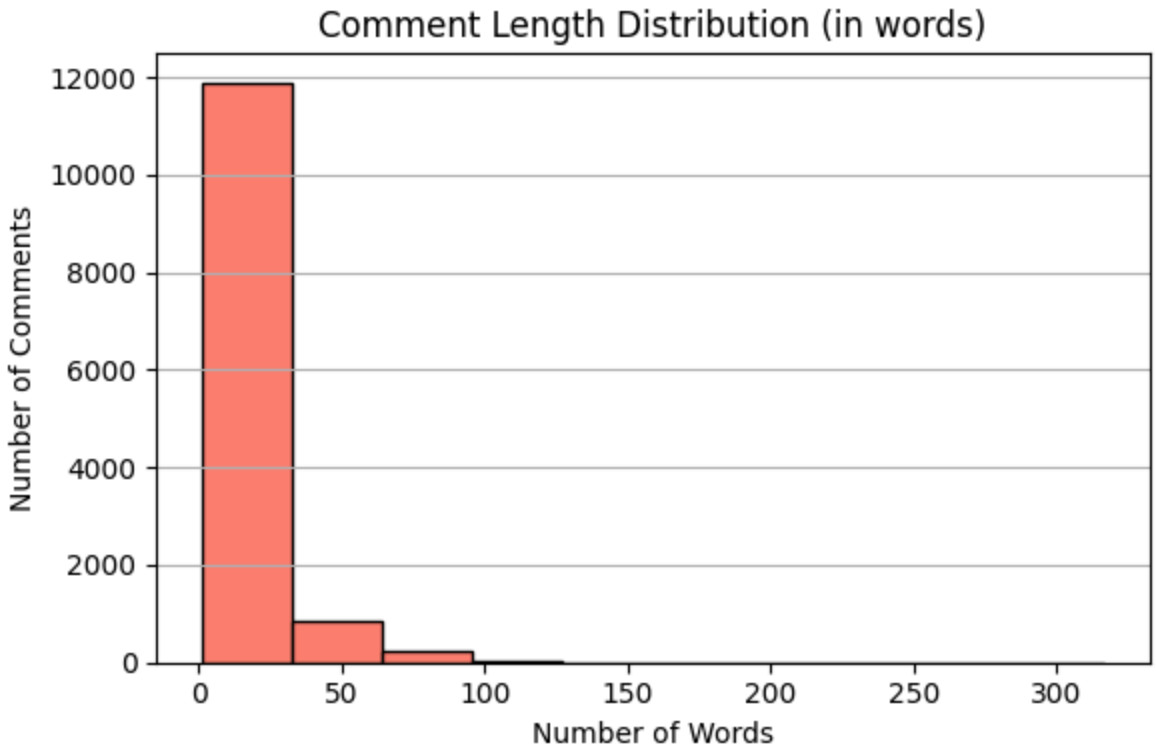
Oversampling minority classes

Undersampling majority classes

Using weighted loss functions during model training

Applying label smoothing techniques

Comment Length Analysis



The comment length distribution provides insights into the textual structure of the reviews:

Most comments are relatively short, with the majority under 50 words

There's a long tail distribution where a few comments exceed 100+ words

The average comment length is approximately 30-40 words

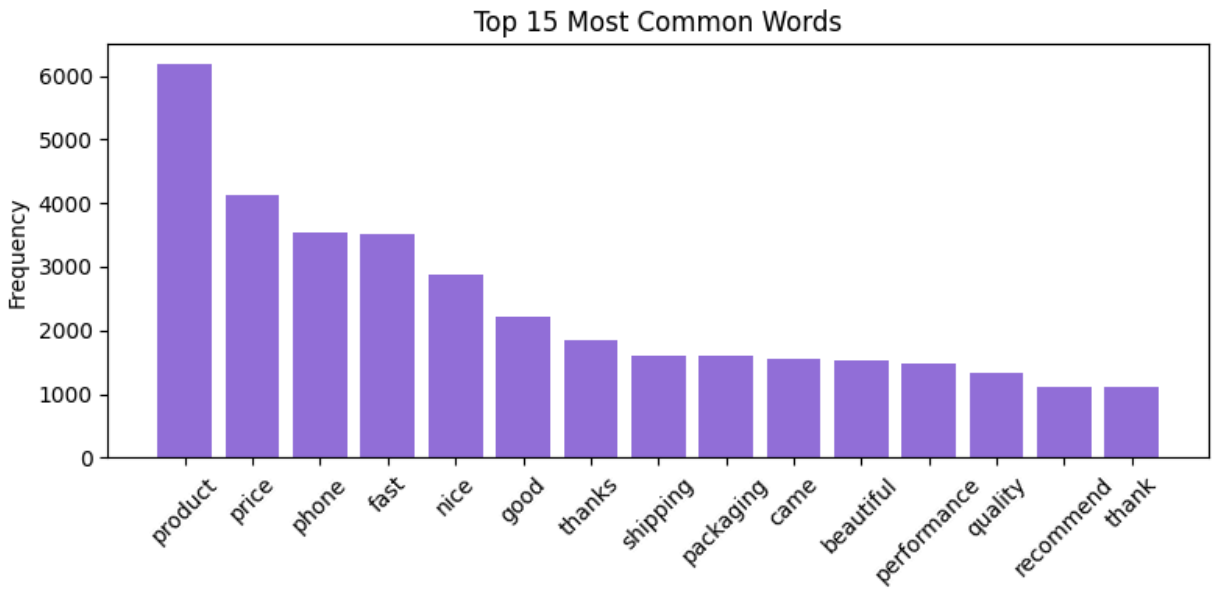
From an NLP perspective, this distribution informs several design decisions:

Padding/truncation strategies can be optimized around the 50-word mark

The short average length benefits real-time inference and reduces computational costs

Models with limited context windows (like CNNs or small Transformer variants) may be sufficient

Word Frequency Analysis



The analysis of the 15 most common words after preprocessing reveals:

High frequency of terms like "product", "price", "phone", "fast", and "nice"

These terms align with the electronics review domain, focusing on product quality and performance

The most frequent words correlate well with the most common labels, showing semantic alignment

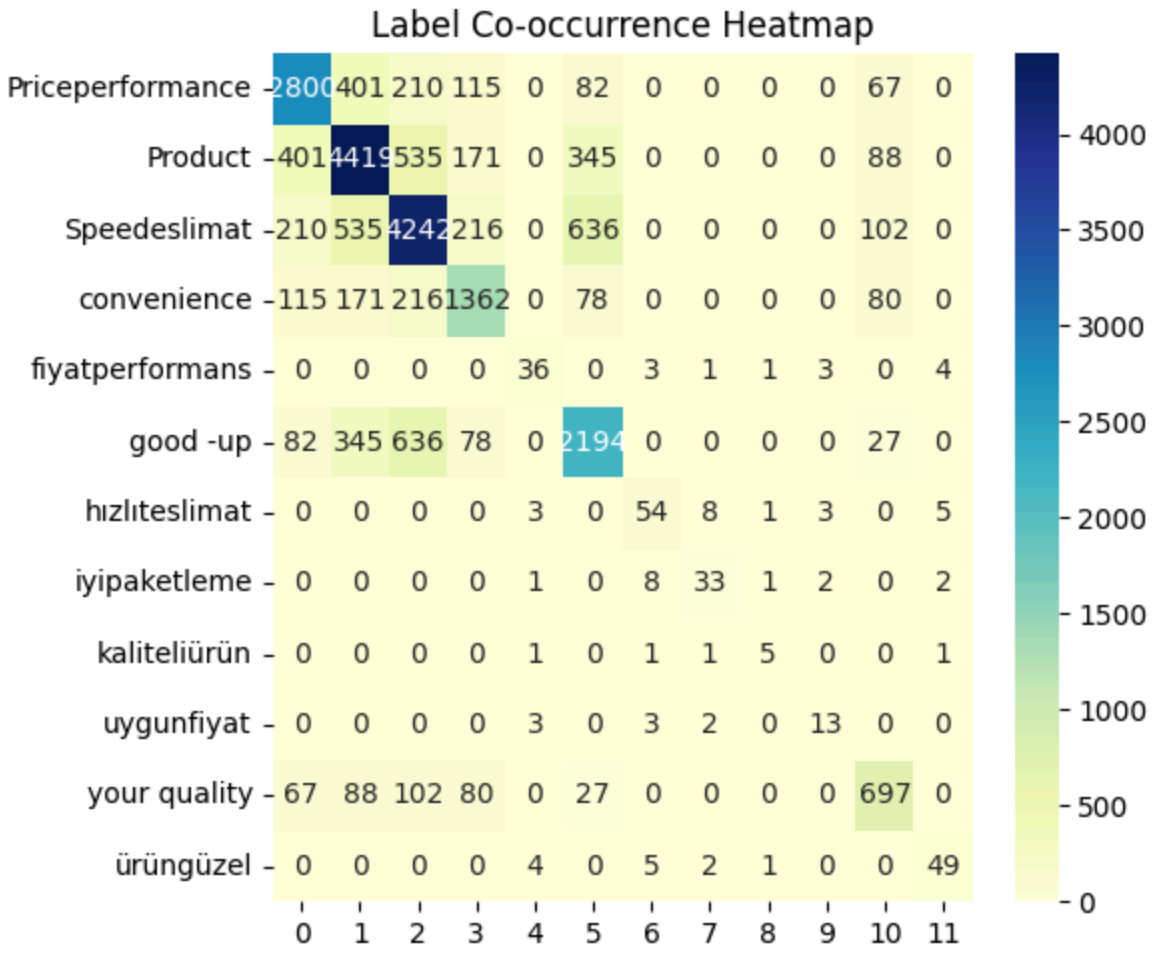
These high-frequency terms are particularly valuable for:

TF-IDF vectorization as they represent core concepts

Keyword-based feature extraction

Understanding the semantic focus of the dataset

Label Co-occurrence Analysis



The label co-occurrence heatmap reveals how frequently different labels appear together in the same comment:

Strong diagonal values indicate that most comments are dominated by a single label

High off-diagonal values between certain label pairs (e.g., "Product" and "Speedslimat", "Priceperformance" and "Product") show significant co-occurrence

This suggests interdependencies between labels that should be considered in multi-label classification approaches

This co-occurrence pattern informs the choice between:

Binary Relevance (treating each label independently)

Classifier Chains (modeling label dependencies)

Label Powerset (treating each combination as a unique class)

Text Embedding Techniques

1. Continuous Bag of Words (CBOW)

The CBOW model (Cbow.py) predicts a target word based on its surrounding context words:

Implemented using Gensim's Word2Vec with sg=0 parameter

Trained with vector size of 100, window size of 5, and minimum word count of 1

Word vectors are converted to PyTorch tensors for compatibility with deep learning frameworks

Document vectors are created by averaging word vectors, providing a simple but effective document representation

CBOW tends to perform well on smaller datasets and is particularly effective at capturing semantic similarities between words that appear in similar contexts. The model learns to predict a word given its surrounding context, which helps it focus on the most common and predictable words in the corpus.

2. Skip-gram Model

The Skip-gram model (skipgram.py) takes the opposite approach to CBOW, predicting context words given a target word:

Implemented using Gensim's Word2Vec with sg=1 parameter

Trained with the same hyperparameters as CBOW for fair comparison

Document embeddings are generated by averaging word vectors

Embeddings are saved in both CSV and PyTorch formats for flexibility in downstream applications

Skip-gram typically performs better on larger datasets and excels at capturing relationships between rare words. By predicting context from target words, it creates more nuanced embeddings that can better represent the semantic space of the reviews.

3. TF-IDF Vectorization

The TF-IDF approach (TF_IDF.py) represents documents based on term frequency and inverse document frequency:

- Implemented using scikit-learn's TfidfVectorizer
- Custom tokenizer applied for consistent preprocessing across all embedding methods
- Special tokens added to mark sentence boundaries (<s> and </s>)
- Results saved as a CSV file with original IDs and labels preserved

TF-IDF provides a statistical measure of word importance that differs from the neural approaches of CBOW and Skip-gram. It emphasizes words that are frequent in a document but rare across the corpus, which can be particularly useful for identifying distinctive terms in each review.

4. SBERT Embeddings

The SBERT-based approach (sbert.py) generates dense vector representations using a transformer-based sentence encoder:

Fine-tuned using all-MiniLM-L6-v2 from SentenceTransformers for domain-specific sentence similarity

Training pairs constructed by grouping comments with identical labels and creating positive pairs

Text preprocessing includes lowercasing, punctuation removal, stopword filtering, and lemmatization

Trained with MultipleNegativesRankingLoss for 20 epochs on meaningful sentence pairs

Final embeddings saved as tensors along with original labels and comment indices

SBERT captures semantic similarities between comments more effectively than frequency-based models like TF-IDF, offering robust representations ideal for downstream tasks like classification or clustering in NLP pipelines.

Comparative Analysis of Embedding Techniques

Each embedding technique offers distinct advantages:

1. **CBOW**:

Faster training time

Better at capturing semantic relationships for frequent words

Smoother word representations due to averaging contexts

2. **Skip-gram**:

Better performance on rare words

More fine-grained semantic distinctions

Generally produces higher quality word vectors at the cost of training efficiency

3. **TF-IDF**:

Captures statistical importance rather than semantic relationships

Sparse representations that can be more interpretable

No training required, making it computationally efficient

The choice between these methods depends on the specific downstream task, dataset size, and computational constraints.

4. **SBERT**:

Captures deep semantic similarity between full sentences

Provides dense, contextualized embeddings suitable for sentence-level tasks

Requires more computational resources but yields superior performance for classification and clustering

Analysis

Analysis revealed a critical vocabulary mismatch between the Skip-gram model's training corpus and the evaluation dataset. When words in the test documents were not present in the model's vocabulary, they were represented as zero vectors, resulting in information loss. This was particularly problematic for domain-specific terminology in electronics reviews.

The multi-label nature of our classification task introduced additional complexity. The standard threshold of 0.5 used for binary decisions proved inappropriate for our label distribution, which exhibited significant imbalance as demonstrated in our exploratory data analysis. This imbalance particularly affected the Skip-gram

and SBERT models, which lacked the statistical robustness of frequency-based approaches like TF-IDF.

The dimensionality of the Skip-gram embeddings (100) proved insufficient to capture the semantic nuances of the electronics review domain. Conversely, SBERT's high-dimensional embeddings (384 dimensions) likely suffered from the curse of dimensionality given our limited training dataset size, leading to sparse representation spaces that complicated classification.

CBOW's fundamental approach of averaging context word vectors to predict target words creates a natural alignment with our document representation strategy, which similarly averages word vectors to create document embeddings. This architectural consistency likely contributed to CBOW's moderate success (50% accuracy) compared to Skip-gram's poor performance, as the averaging mechanism preserves the distributional properties that CBOW was specifically trained to capture.

Project Structure

The project is organized as follows:

src/ : Contains all source code files

data_preprocessing.py : Handles dataset translation

data_vis.py : Performs exploratory data analysis

Cbow.py : Implements CBOW embeddings

skipgram.py : Implements Skip-gram embeddings

TF_IDF.py : Implements TF-IDF vectorization

datasets/ : Stores all data files

interim/ : Contains intermediate processed data

interim/embeddings/ : Stores generated embeddings

figures/ : Contains visualization outputs

label_distribution.png : Shows frequency of each label

comment_length_distribution.png : Displays distribution of comment lengths

top_15_most_common_words.png : Visualizes most frequent words

label_cooccurrence.png : Heatmap of label co-occurrences

models/ : Stores trained models

Conclusion

This project demonstrates a comprehensive NLP pipeline for analyzing non-English text data:

1. **Data Preparation**: Successfully translated Turkish electronics reviews to English, making them accessible for English-based NLP techniques. The preprocessing steps effectively standardized the text while preserving semantic content.

2. **Data Understanding**: Exploratory analysis revealed important patterns:

Significant class imbalance in label distribution

Predominantly short comments with a long-tail distribution

Clear semantic focus on product quality and performance

Meaningful co-occurrence patterns between labels

3. **Text Representation**: Implemented three different embedding techniques that capture complementary aspects of the text:

CBOW: Captures semantic relationships by predicting target words from context

Skip-gram: Captures semantic relationships by predicting context from target words

TF-IDF: Captures statistical importance of words in documents

These embeddings provide different perspectives on the text data and can be used for various downstream tasks such as classification, clustering, or recommendation systems.

The insights gained from this analysis suggest several directions for future work:

Addressing class imbalance through sampling or weighting techniques

Exploring hybrid embedding approaches that combine statistical and neural methods

Implementing multi-label classification models that account for label dependencies

Fine-tuning pre-trained language models like BERT or RoBERTa on this domain-specific data

The project successfully demonstrates the application of modern NLP techniques to non-English data, highlighting the importance of proper preprocessing and the versatility of different embedding approaches in capturing the semantic richness of user-generated content.

Future Scope

In the upcoming phases of this project, we plan to expand both the breadth and depth of our analysis with the following goals:

We would like to explore the following methods as well

Recommended Further Models

1. **Transformer-Based Classification Models**
2. **Graph Neural Networks for Label Dependencies**
3. **Ensemble Methods Combining Multiple Embeddings**
4. **Attention-Based Multi-Label Classification**
5. **Hierarchical Classification Model**

Clustering-Based Emotion Analysis and Semantic Structuring

While this project focused on generating and comparing various text embeddings, a promising direction for future work involves clustering these embeddings to uncover latent structures such as emotional patterns in the review data. By applying unsupervised learning techniques like K-Means Clustering, we aim to group similar review embeddings together based on their semantic closeness. The hypothesis is that reviews expressing similar emotions or sentiments will cluster together. This clustering-based analysis provide insights into the semantic grouping of reviews but may also support multilingual generalization of emotional tone, especially valuable given the original dataset's Turkish origin. This will allow us to:

Discover dominant emotional themes

(e.g., satisfaction, frustration, excitement) without relying on predefined labels.

Validate the consistency and quality of the embeddings

—if the clusters align well with known sentiment labels or emotion categories, it confirms that the embedding model effectively captures nuanced semantics.

Create a basis for emotional summarization or **emotion-aware recommendation systems**, especially useful in e-commerce settings where customer feedback often reflects emotional experiences with products.

Clustering and Emotion Analysis

We intend to perform unsupervised clustering on the embeddings (TF-IDF, Word2Vec, SBERT, etc.) to explore the emotional patterns present in the reviews.

Techniques like K-Means clustering will be used to group reviews based on semantic similarity.

Post-clustering, we will analyze whether these clusters correspond to specific emotions or sentiments expressed in the text.

This analysis will help assess how well the embeddings capture underlying emotional tones and could be extended to emotion-based summarization or feedback classification.

Translation and Embedding of womanComments.csv

The final untranslated dataset, womanComments.csv , will be translated from Turkish to English.

After translation, we will generate all types of embeddings for this dataset as well, including:

TF-IDF

Word2Vec (CBOW and Skip-gram)

Fine-tuned SBERT

These embeddings will be stored and used for both clustering and classification tasks.

Extended Model Evaluation

Beyond the Random Forest classifier already used, we will evaluate the effectiveness of each embedding on two additional models.

One of the planned models will be a neural network to explore deep learning-based classification.

This will help us understand the strengths and limitations of traditional versus deep learning methods for text classification in this domain.

All models will be compared based on metrics like accuracy, precision, recall, F1-score, and training time.