FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Mkrtich Hovsepyan

Deep Learning Models for Product Mapping

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: RNDr. Kateřina Macková

Study programme: Computer Science with specialization in Artificial Intelligence

Prague 2024

I declare that I carried out this bachelor thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

To my supervisor, RNDr. Kateřina Macková, whose support, guidance, and encouragement have been invaluable throughout this journey.

Title: Deep Learning Models for Product Mapping

Author: Mkrtich Hovsepyan

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Kateřina Macková, Department of Theoretical Computer Science and Mathematical Logic

Abstract: The thesis addresses the challenge of matching products from different e-commerce platforms called Product Mapping. The problem in Product Mapping is to decide whether two products from various sources refer to the same product. The main goal of this thesis is to evaluate different deep-learning approaches to increase the accuracy of Product Mapping techniques. We use all available textual and image data that can be extracted from e-commerce platforms to find the most suitable models techniques. We experiment with four different datasets: ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart. We compare models such as TF-IDF, Word2VEc, and BERT-based transformers for text preprocessing and we use CNNs like GG16, ResNet50, Inception V3, and EfficientNet for image preprocessing. Then, we use machine learning, and deep learning classifiers for computing similarity scores for individual features. We obtained promising results with BERT-based architectures for text data and multimodal models for mage data. These methods improve accuracy and F1 score, achieving superior results on the datasets. Which highlight the critical role of deep learning techniques in advancing the field of e-commerce product mapping.

Keywords: Product Mapping, Product Matching E-commerce Platforms, Deep Learning Techniques, Textual Data, Image Data, TF-IDF, Word2Vec, BERT, Convolutional Neural Networks, VGG16, ResNet50, Inception V3, EfficientNet, Machine Learning, Logistic Regression, Support Vector Classifier, Multi-Layer Perceptron (MLP), Large Language Models, ProMap datasets, Amazon-Google Dataset, Amazon-Walmart Dataset

Contents

Introduction

Product mapping is closely related to linking, mapping, and matching products in multiple databases or catalogs. It allows one to discern the same product within several systems, even in cases where the description, image, etc., of the product are different. This is a comparison method based on textual and visual data that provides for relating identical objects, which can be presented in different manners in different e-commerce environments. Product mapping ensures the consistency of data and a good user experience of using Computer Vision and Natural Language Processing which enables to correctly compare prices and manage inventories to help customers find what they want on marketplaces.

In this thesis we work with four datasets: ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart. The datasets comprise names and descriptions; the ProMap datasets also include images of the products and more textual features. The thesis compares several methods and finds the best model for English and Czech data. We deploy various models to pre-process all the text and image features characterizing each product such as TF-IDF, Word2Vec, BERT-based transformers for text data, and Convolutional Neural Networks with models VGG16, ResNet50, Inception V3, and EfficientNet for image data.

Afterwards, the machine and deep learning models classify characteristics extracted in the previous step. We extract the similarity score of the characteristics between two products based on comparing their names, short descriptions, specifications, long descriptions, and images. Apart from this, Multi-Layer Perceptron, Logistic Regression, Random Forest, Naive Bayes, KNN, and several other classifiers were trained. The outcomes of our experiments improved accuracy and F1 score for Product-Mapping tasks through fine-tuning BERT-based architectures and multimodal models. These techniques were applied to achieve amazing progress on datasets ProMapEn and ProMapCz. The best F1 scores obtained were 0.73, 0.72, 0.93, and 0.99 for ProMapEn, ProMapCz, Amazon-Google and Amazon-Walmart, respectively.

We test the model performance when using only the name feature against all available textual features combined. This might help us understand the significance of the name feature. the contribution of the name feature and how it influences the overall performance of the Product Mapping models. Our experiments can decide whether the name feature alone can give accurate product mapping or if substantial increments in accuracy can be achieved with other textual features.

1 Product Mapping

1.1 Product Mapping Task

Product Mapping is one of the critical aspects of the e-commerce industry: linking, mapping, and matching products among different databases or catalogs. The process ensures consistency in identifying the same products across multiple systems, where descriptions, images, and other features could differ. It, also, ensures data integrity, a better customer experience, and that prices and stocking are matched correctly across all these multiple marketplaces.

Product Mapping contrasts text and visual information to draw a correct relationship between similar products shown differently on various e-commerce platforms. For instance, the same smartphone could be described or have different images and specifications to be sold on Amazon, Walmart, or Google Shopping. Product Mapping identifies and matches those slight variations with a single product identity.

1.1.1 The Importance of Product Mapping

Consistency in data: One of the goals that Product Mapping aims at is to ensure consistency in product data. Consistency in data benefits both the seller and the buyer. Sellers can see a potential sale because information about their products is inconsistent in other places, whereas buyers can make an informed choice because they find information consistent.

User Experience: User experience is at the heart of the success of e-commerce operations. When the product data are consistent, the customers should easily be able to compare products on a different scale. Accurate mapping allows the customer to avoid discrepancies that could raise questions of trust and frustration.

Accurate Price Comparisons: Comparative pricing is one of the major features of online shopping. Product Mapping enables customers to make an accurate price comparison, thus enabling them to compare products like for like. This is helpful, especially in a highly competitive market, where a slight price difference can alter the consumers' buying decisions.

Inventory Management: With effective Product Mapping, inventory management becomes an easy task. A retailer can track products through different platforms and ensure that the stock level is always updated in real time. This removes the issue of items available on one site but out of stock on the other, which brings customer dissatisfaction and loss of sales.

1.1.2 Techniques in Product Mapping

Product Mapping, in general, contains several advanced techniques like Natural Language Processing and Computer Vision technologies, which are mainly applied to functions like comparing and matching one product to another.

Natural Language Processing (NLP): NLP works with textual data linked to the products. Generally, an analysis contains the product's name, description, specifications, and reviews. An efficient matching process will be conducted

through it, wherein the textual data identified using NLP techniques will find similarities and differences.

We deal with this problem by considering not only all available textual data but also experiments in understanding the influence of various features. For instance, we could compare the performance between models using only the name feature of the product and all available textual features combined. Such comparison may provide insight into the role or importance of the name feature in this kind of Product Mapping and help understand how much it influences the whole performance of the models. Considering its importance in making mapping more accurate, we will see this name feature alone. **Computer Vision**: Computer Vision is important for products that have a high dependence on their visual features—for example, clothing or electronic items. This technology helps compare images of a product and looks into the similarity of the visual representation. Comparisons made herein could be based on color, shape, or even design features to say if the two products are similar.

Hybrid Approaches: Most frequently, the combinations of approaches offer the best efficacy in Product Mapping. Using both textual and visual data provides a proper understanding of the products, thus allowing for a better match.

The real future of Product Mapping lies in the advancement of AI and machine learning. Systems that are more accurate and perpetually innovated for such matters will soon be able to handle much more sophisticated forms of variation in product data. The two areas where further improvement is possible are handling more intricate product data variation and integrating real-time data.

1.2 Datasets

1.2.1 Existing Datasets

There are several datasets available for Product Mapping, including:

- **WDC Dataset**: Created by Web Data Commons [1], it contains only product names without additional details like descriptions, specifications, or images.
- **Amazon-Walmart Dataset [2]**: This dataset has detailed product information but includes only distant non-matches.
- **Amazon-Google Dataset [3]**:Contains product names, descriptions, and prices, but lacks specifications, images, and non-matches.
- **Abt-Buy Dataset [4] [5]**: Does not have comprehensive product information.
- **Other Datasets**: Such as DBLP-ACM, DBLP-Scholar, Shopmania, and Amazon Review Data, [4] [5] which are not specifically designed for Product Mapping tasks.

1.2.2 Chosen Datasets

For this thesis, we selected three datasets: ProMap dataset collection, Amazon-Google, and Amazon-Walmart.

- **ProMap [6]** : Chosen for its comprehensive coverage of all available product details, including images and close non-matches.
- **Amazon-Google [3]**: Selected for its well-documented matching pairs.
- **Amazon-Walmart [2]**: Included for its detailed product information, despite having only distant non-matches.

These datasets were selected to ensure a robust evaluation of Product Mapping models with a variety of product details and match types.

1.2.3 ProMap Datasets

The ProMap dataset collection [6] consists of two primary datasets, ProMapCz and ProMapEn which contain product pairs from Czech and English e-shops, respectively. These datasets are significant in our study, helping us to develop deep-learning models and compare them. Every data pair has two products, and each product has text and image features. The text features for each product are "name," "short_description," "long_description," and "specification." Each product has multiple images associated with it. Human annotators created Both datasets manually, ensuring their high quality.

ProMapCz

The ProMapCz dataset [6] consists of 1,495 pairs of Czech products, including both matching and non-matching pairs. The products are from Alza.cz and Mall.cz. It includes 706 unique products from Alza.cz and 1,409 from Mall.cz, with 504 matches, 456 close non-matches, and 535 medium non-matches. The dataset is divided into a training set of 1,196 pairs and a testing set of 299 pairs based on an 80:20 split. Matches are pairs referring to the same product, close non-matches are similar but not the same product which are slightly more different than close non-matches.

ProMapEn

The ProMapEn dataset [6] has 1,555 English product pairs from Walmart.com and Amazon.com, having 1,555 unique products from Walmart.com and 751 from Amazon.com. There are 500 matches, 509 close non-matches, and 537 medium non-matches. This dataset is divided into train and test sets, with 1,244 pairs for training and 311 pairs for testing, having an 80:20 ratio.

Amazon-Google and Amazon-Walmart Datasets

In addition to the ProMap datasets, we use datasets from Amazon-Google and Amazon-Walmart. These datasets provide additional diversity and information, helping to ensure the unbiasedness of our models in different datasets. They are

larger than ProMap datasets but do not provide several levels of non-matching products, do not contain image data, and have fewer text features. Original datasets are too large and strongly unbalanced. Therefore, we used shrunk and balanced versions, which were created to make them more similar to ProMap datasets [6].

Amazon-Google Dataset

The Amazon-Google [3] dataset consists of product pairs from Amazon and Google Shopping. Each data point consists of two products, and each product has text features ("name," "short description," "long description"), price, and manufacturer. The dataset has 1,363 products from Amazon, 3,226 from Google, and 1363 matches and 1935 non-matches, and the data is split 80:20 into training and testing datasets, which created 2589 pairs in train and 648 pairs in the test dataset.

Amazon-Walmart Dataset

The Amazon-Walmart dataset [2] includes product pairs from Amazon and Walmart. The dataset has 24,583 different products. Each product has a title, brand, short and long descriptions, URLs, etc. There are 1154 matching pairs and 2000 non-matches, and the data is split 80:20 into training and testing datasets, which created 2515 pairs in the train and 630 in the test dataset.

1.3 State of the Art

1.3.1 Traditional Approaches

Originally, the Product Mapping techniques were based on some string similarity measurements like cosine similarity, edit distance, and Jaccard similarity. They relied heavily on textual information to do a line-by-line comparison. Since the product descriptions are wildly variant and usually relatively sparse, the methods proposed in the past usually gave inaccurate results.

Deep NLP Models

Transformer-based Models: The transformer models, especially BERT, have addressed NLP and Product Mapping problems. With its other derivations, such as Sentence-BERT, BERT helps provide embeddings of product titles and descriptions to make the matching more accurate.

Sentence-BERT: In contrast, Sentence-BERT leverages siamese and triplet networks over the pre-trained BERT model into sentence embeddings. This approach can thus capture context adequately from within the product descriptions and demonstrates good performance compared to the conventional string similarity measures [7].

Deep Learning Architectures for Multimodal Data

Text and Image Similarity: It has been shown that using a multimodal approach to combine data in the form of text and image information with deep neural networks has improved product matching accuracy. The up-to-date models with embedded images and text better match products described by both modalities.

1.3.2 State of the Art Models

These sophisticated datasets and models introduced in recent studies help acquire more advanced Product Mapping features. For instance, the ProMap datasets, including ProMapCz and ProMapEn, have complete information concerning the products; they contain images of products and their textual descriptions. The best results were obtained on these datasets with neural network-based models, achieving F1 scores of 0.77 on ProMapCz and 0.70 on ProMapEn. [6].

Comparison of Models

The ProMap datasets have been created to overcome the inadequacies of existing datasets, which generally comprise far-away non-matching pairs and have incomplete information about products. Including close and medium non-matching products makes them, in a way, a more realistic benchmark and, hence, challenging for testing Product Mapping models. Below is a table showing various machine learning methods tested on several datasets: ProMapEn, ProMapCz, Amazon-Google and Amazon-Walmart. Then, the models trained are evaluated with F1 score, Precision and Recall. The best results in every dataset attained were the models trained on text and image processing methods and also on deep learning techniques. The texts are preprocessed by removal of unwanted characters, lemmatization, lower casing, and turning them into numerical vectors with TF-IDF. This step is then followed up with cosine similarity computation, making it possible for text processing. As far as image processing is concerned, resizing is done, after which they are changed into grayscale. Object detection is done through edge detection and bounding box techniques, with perceptual hash generation in computation for the image similarity measure. During this work, many machine learning models were trained and tuned through Grid and Random Search: Logistic Regression, Support Vector Machines, Decision Trees, Random Forest, Neural Networks. Results indicated that the neural network-based model fared very well with every dataset: close-to-high F1-scores and high-precision/recall results confirm its suitability for handling text data or image data in product mapping tasks.

Dataset	**F1**	**Precision**	**Recall**
ProMapEn	0.70	0.71	0.70
ProMapCz	0.77	0.79	0.76
Amazon-Google	0.99	1.00	0.98
Amazon-Walmart	0.93	0.90	0.96

Table 1.1 Comparison of Machine Learning Methods on Various Datasets

1.4 Machine Learning and Deep Learning Models Used in the Thesis

We have used multiple classifiers to predict whether the product pair is a match. The following are the classifiers used:

- **XGBoost**
  - **XGBoost** [8] is an optimized distributed gradient boosting library, aiming at being highly efficient, flexible, and portable. A machine learning library interfaces many languages under the Gradient Boosting framework. XGBoost provides parallel tree boosting, also known as GBDT or GBM, a topic that tops Kaggle leaderboards in most cases when solving data science problems quickly and efficiently. The model was set with a maximum depth of 10 and 1000 estimators for our use.
- **Logistic Regression**
  - **Logistic Regression** [9] is a statistical analysis method used when the data consists of one or more independent variables determining an outcome. The outcome is measured by a dichotomous variable, having only two possible outcomes. In other words, it is a method for predicting the probability of a binary response based on one or more predictor variables.
- **Random Forest**
  - **Random Forest** [10] is an ensemble learning method that builds multiple decision trees during training and outputs the most common class for classification tasks. The model used in this thesis includes 1000 trees with a maximum depth of 10, which enhances prediction accuracy and reduces overfitting by averaging the results of multiple trees.
- **SVC**
  - **Support Vector Classifier (SVC)** [11] is a specific implementation of the Support Vector Machine algorithm built for classification tasks. Again, an SVC is an SVM for classification and looks for the best hyperplane to find separation between classes of data points. While the terms "SVC" and "SVM" have been used interchangeably, if a person mentioned an "SVC," he referred to this variant variable of the algorithm in classifying. We use an RBF kernel with parameters C=1 and gamma=1.
- **KNN**
  - K-nearest neighbors (KNN) [12] algorithm is generally abbreviated as a k-nn, and it's a way of classifying data that estimates the likelihood of a given point in the data being part of one group or another based on what group the data points closest to it belongs.

- **Naive Bayes**
  - **Gaussian Naive Bayes classifier** [13] is based on Bayes' theorem, assuming independence between every pair of features. This probabilistic classifier is particularly effective for high-dimensional data and is simple to implement, making it a fast and efficient method for classification tasks
- **MLP (Multi-layer Perceptron)**
  - Multi-layer Perceptron [14] is also known as MLP. It means fully connected dense layers transforming any input dimension to the desired one. It is a neural network that contains multiple acquired layers joined with one another. This can be done by combining neurons together so that the outputs of some neurons become inputs for other neurons to create a neural network.
  - **MLP 1000**
    - hidden_layer_sizes: (1000)
    - max_iter: 1000
    - activation: relu
    - solver: lbfgs
    - learning_rate: adaptive
    - learning_rate_init: 0.01
    - early_stopping: True
    - n_iter_no_change: 1000
  - **MLP 500 500**
    - hidden_layer_sizes: (500, 500)
    - max_iter: 1000
    - activation: relu
    - solver: lbfgs
    - learning_rate: adaptive
    - learning_rate_init: 0.01
    - early_stopping: True
    - n_iter_no_change: 1000
  - **MLP 300 200 100**
    - hidden_layer_sizes: (300, 200, 100)
    - max_iter: 1000
    - activation: relu
    - solver: lbfgs
    - learning_rate: adaptive
    - learning_rate_init: 0.01
    - early_stopping: True
    - n_iter_no_change: 1000

  - **MLP 1000 tanh**
    * hidden_layer_sizes: (1000)
    * max_iter: 1000
    * activation: tanh
    * solver: lbfgs
    * learning_rate: adaptive
    * learning_rate_init: 0.01
    * early_stopping: True
    * n_iter_no_change: 1000
  - **MLP 1000 relu adam**
    * hidden_layer_sizes: (1000)
    * max_iter: 1000
    * activation: relu
    * solver: adam
    * learning_rate: adaptive
    * learning_rate_init: 0.01
    * early_stopping: True
    * n_iter_no_change: 1000
  - **MLP 1000 relu 0.001**
    * hidden_layer_sizes: (1000)
    * max_iter: 1000
    * activation: relu
    * solver: lbfgs
    * learning_rate: adaptive
    * learning_rate_init: 0.001
    * early_stopping: True
    * n_iter_no_change: 1000

- **AdaBoost**
  - AdaBoost (Adaptive Boosting) [15] is an ensemble learning algorithm that concatenates many weak classifiers to get one robust classifier. Successive rounds allow the algorithm to adjust weights according to the previous classifier's performance, emphasizing those instances misclassified in that round. Each base weak model is weighted by its performance and combined into the final model. For the number of estimators, we have used 1000, and the learning rate used here is 0.01.

- **Gradient Boosting**
  - Gradient Boosting [16] is a very powerful machine learning technique that builds a predictive model in stages. It basically involves the creation and summation of weak predictive models, typically decision trees, one at a time, in such a way that each new model makes up for

the shortcomings of previous ones. This process uses calculus' concept of gradients to minimize errors. Hence the name 'gradient boosting.' We use the model with 1000 estimators, a learning rate of 0.01, and a maximum depth of 10.

- **Extra Trees**
  - Extra trees: Extra trees [17] stand for extremely randomized trees. It forms another ensemble supervised machine learning method using decision trees used by the Train Using AutoML tool. The extra trees algorithm, similar to random forests, generates many decision trees. However, in extra trees, the sampling for every tree is random, and there is no replacement. We use 1000 estimators and a maximum depth of 10.
- **Bagging**
  - Bagging (Bootstrap Aggregating) [18] is the ensemble learning method generally used to reduce variance in a noisy data set. In bagging, a random sample is made out of data in a training set; this is done with replacement, meaning that individual data points may be selected more than once. In this implementation, SVC is used as the base estimator with 100 estimators
- **Linear SVC**
  - Linear Support Vector Classifier [11] is a type of machine learning algorithm used in classification problems. This works by using a straight line or a hyperplane that separates data points of different classes. If the given data can be precisely linearly separable, then only linear SVMs can be applied.
- **SGD**
  - Stochastic Gradient Descent (SGD) [19] is an algorithm for the optimization of an objective function with adequate smoothness properties, such as differentiability or subdifferentiability; it is iterative by nature. It may also be understood as a stochastic approximation of optimization by gradient descent: it replaces the real gradient of the objective function on the basis of a large but finite sample of the data set with a gradient estimate defined on the basis of a randomly chosen subset.

2 Comparison of Image Representation Models

2.1 Image Preprocessing Methods

In the ProMapEn and ProMapCz datasets, every product has several images. The images show the product from different angles, give an overview of some features, and provide important product characteristics that cannot be captured in the text data. Therefore, they are a vital data source for the product mapping task.

2.2 Data Processing Approach

Our methodology for dealing with image data consists of three parts and is outlined as follows:

- **Image Preprocessing**: First, we take resizing, normalizing, and other known image processing methods to each image:
  - **Resizing**: Each image is resized so that the shorter side is 256 pixels, preserving the aspect ratio (ensures that the shorter side of the image is consistently resized to a specific dimension).
  - **Center Cropping**: From the resized image, a central square patch of 224x224 pixels is extracted (the pretrained models like ResNet, VGG, and EfficientNet are trained on 224x224 pixel images.).
  - **Conversion to Tensor**: The image is converted into a multidimensional array format suitable for input into neural networks.
  - **Normalization**: The mean and standard deviation values are used to normalize the image tensor; these are dataset-dependent values on which the pre-trained models were initially trained. As done earlier in this chapter, this normalization step will standardize input data across different images and even across datasets.
- **Feature Extraction**: Then we use pretrained on freely available image data models for feature extraction. We tried VGGm ResNet, EfficentNet, and Inception.
- **Similarity Computation**: We compute the similarity scores by comparing every possible pair of images between the two products. We chose cosine similarity for this part, which showed better results than other distance metrics. Given n images for one product and m images for another, this results in $n \times m$ matrix with similarity scores.
- **Max Similarity Selection**: From the computed similarity scores, we select the highest value. This approach is based on the observation that the same products probably have at least one pair of highly similar images, even if

other pairs have low similarity scores. Selecting the maximum value makes it easier to perform the comparisons by focusing on the most relevant image pairs.

The maximum similarity score is then used as a feature in subsequent machine learning and deep learning models for product mapping tasks.

The following figure illustrates our image preprocessing and comparison architecture:

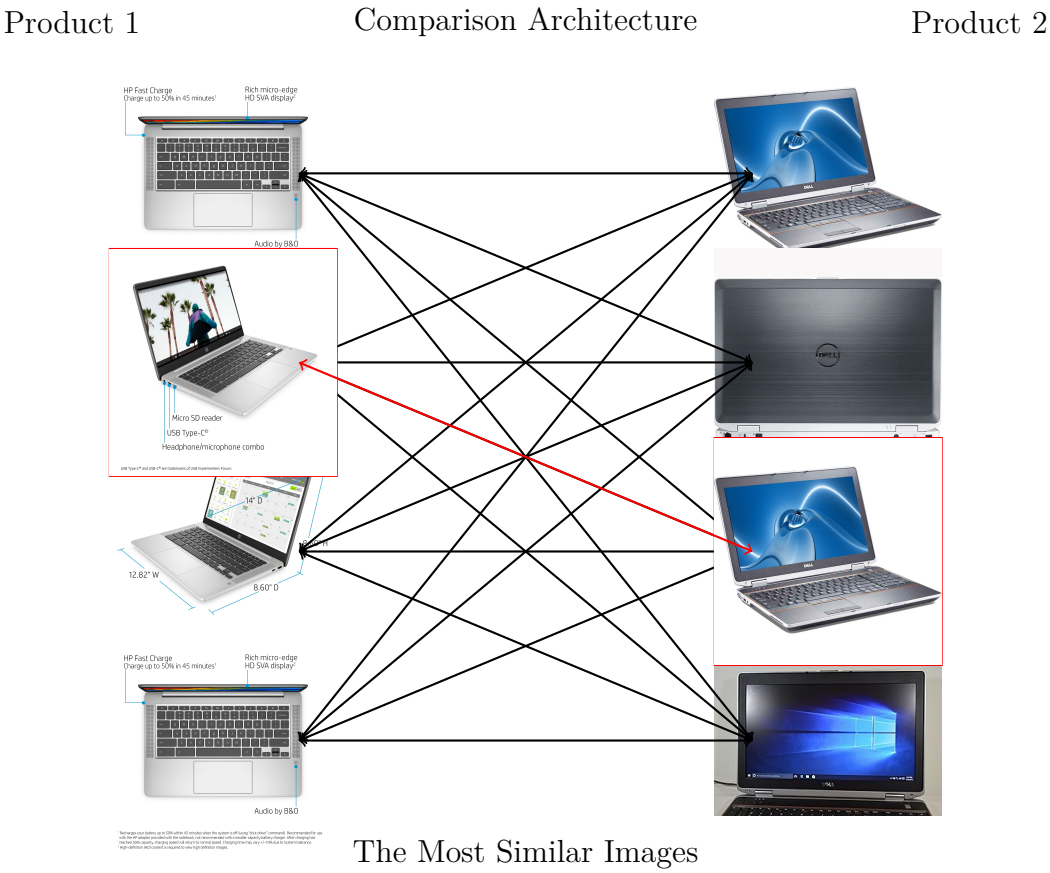

Figure 2.1 Comparison of product images with double-sided arrows indicating comparisons and highlighting the most similar pairs.

2.2.1 Feature Extraction Models

We used the following methods for the feature extraction

- **VGG16**
- **ResNet50**
- **Inception V3**
- **EfficientNet**
- **Custom Model - MaxOutputEnsemble**

Every model catches different features from the images. This gave the idea to make an ensemble model combining all of the methods together (MaxOutputEnsemble). EfficientNet and MaxOutputEnsemble showed the best results after performing all experiments.

2.2.2 VGG16

VGG16 [20] is a classical convolutional neural network pre-trained on ImageNet dataset [21], which is a freely available large database of images that are designed for different image recognition tasks. The advantage of using this method was that it is simple to understand and can provide good baseline results for product mapping tasks.

Architecture of the VGG Model

The architecture of VGG can be divided into several key components:

- **Convolutional Layers**: 13 convolutional layers with 3x3 filters.
- **Activation Function**: Relu (Rectified Linear Unit)
- **Pooling Layers**: 5 max-pooling layers.
- **Fully Connected Layers**: 3 fully connected layers.
- **Final Layer**: 1 softmax layer

Performance with MLP Classifier We performed several experiments. We involved VGG16 for getting individual image embeddings and afterwards processed in our pipiline described in Data Processing Approach. Then we run several ML models to train binary to detect matching and non-macthcing products. The best result for the ProMapEn dataset using the VGG16 feature extractor combined with the MLP model. The highest F1 score is 0.56 for ProMapEn (while only the image data is used)

Model	F1	Accuracy	Precision	Recall
MLP 1000 tanh	0.56	0.65	0.47	0.7
Logistic Regression	0.55	0.65	0.47	0.64
AdaBoost	0.54	0.64	0.46	0.65
MLP 500 500	0.54	0.66	0.48	0.61
Naive Bayes	0.53	0.63	0.45	0.65
MLP 1000 relu adam	0.53	0.69	0.53	0.52
MLP 1000 relu 0.001	0.52	0.63	0.45	0.62
MLP 1000	0.52	0.63	0.45	0.62
MLP 300 200 100	0.52	0.61	0.43	0.64
Gradient Boosting	0.51	0.5	0.37	0.8
Extra Trees	0.5	0.58	0.4	0.65
XGBoost	0.49	0.61	0.43	0.58
SVC	0.49	0.32	0.32	1.0
Random Forest	0.47	0.55	0.38	0.62
Bagging	0.46	0.73	0.65	0.36
KNN	0.43	0.53	0.35	0.54
Linear SVC	0.0	0.68	0.0	0.0
SGD	0.0	0.68	0.0	0.0

Table 2.1 Detailed results for VGG16 model on ProMapEn dataset using several machine learning algorithms.

We performed the same experiments for the Czech dataset. The results for the ProMapCz dataset were similar, with an F1 score of 0.54. We obtained the best score using the same setup - MLP with VGG16. Detailed results for the ProMapCz textual data are as follows:

Model	F1	Accuracy	Precision	Recall
MLP 1000 tanh	0.54	0.64	0.46	0.66
MLP 300 200 100	0.54	0.68	0.5	0.59
Logistic Regression	0.54	0.64	0.46	0.66
Naive Bayes	0.54	0.64	0.46	0.66
MLP 500 500	0.54	0.67	0.5	0.59
MLP 1000 relu 0.001	0.53	0.65	0.47	0.61
MLP 1000	0.53	0.65	0.47	0.61
Extra Trees	0.52	0.45	0.36	0.91
XGBoost	0.52	0.49	0.38	0.84
MLP 1000 relu adam	0.51	0.5	0.37	0.79
Gradient Boosting	0.51	0.39	0.35	0.98
Random Forest	0.51	0.46	0.36	0.87
AdaBoost	0.49	0.33	0.33	1.0
Bagging	0.49	0.69	0.54	0.45
SGD	0.49	0.42	0.34	0.85
KNN	0.48	0.53	0.38	0.66
SVC	0.38	0.7	0.58	0.29
Linear SVC	0.0	0.67	0.0	0.0

Table 2.2 Detailed results for VGG16 model on ProMapCz dataset using several machine learning algorithms.

2.2.3 ResNet50

ResNet50 [22] or Residual Network with 50 layers, belongs to the family of ResNets, which introduced residual learning to raise the performance related to vanishing gradient while training deep neural networks.

Architecture of ResNet

The ResNet architecture utilizes residual blocks. Rather than learning an unreferenced function for the desired output mapping, ResNet layers focus on learning residuals—the difference between the input and the output.

Key Components

- **Convolutional Layers**: Feature extraction layers.
- **Batch Normalization**: Applied after each convolution to normalize the feature maps.
- **Activation Function**: ReLU (Rectified Linear Unit).
- **Residual Connections**: Skip connections that add the input of the residual block directly to its output.

Results

Results on ProMapEn: When combined with an MLP model, ResNet50 achieved notable results on the ProMapEn dataset. The highest F1 score recorded

was 0.60. Below is the MLP classifier 1.4. The F1 score for the ProMapEn dataset is 0.60. Detailed results for the ProMapEn dataset are as follows:

Model	F1	Accuracy	Precision	Recall
MLP 1000 relu adam	0.6	0.73	0.57	0.64
MLP 1000 tanh	0.59	0.7	0.54	0.66
SVC	0.59	0.68	0.5	0.71
Naive Bayes	0.59	0.7	0.53	0.66
MLP 500 500	0.59	0.68	0.51	0.7
Logistic Regression	0.59	0.7	0.54	0.66
MLP 300 200 100	0.58	0.67	0.49	0.7
AdaBoost	0.58	0.64	0.46	0.78
MLP 1000	0.57	0.65	0.48	0.72
MLP 1000 relu 0.001	0.57	0.65	0.48	0.72
Extra Trees	0.55	0.54	0.41	0.87
Gradient Boosting	0.54	0.53	0.4	0.84
Bagging	0.53	0.73	0.62	0.47
XGBoost	0.53	0.58	0.41	0.74
Random Forest	0.53	0.54	0.4	0.81
KNN	0.52	0.6	0.43	0.65
Linear SVC	0.0	0.68	0.0	0.0
SGD	0.0	0.68	0.0	0.0

Table 2.3 Detailed results for ResNet50 model on ProMapEn dataset.

Results on ProMapCz: We performed the same experiments for the Czech data. The 0.54 F1 score for the ProMapCz dataset was obtained as the best score by using the MLP classifier 1.4. Detailed results for the ProMapCz dataset are as follows:

Model	F1	Accuracy	Precision	Recall
MLP 1000 tanh	0.54	0.64	0.47	0.64
Random Forest	0.54	0.52	0.39	0.87
AdaBoost	0.54	0.69	0.52	0.56
MLP 1000	0.53	0.66	0.49	0.57
MLP 500 500	0.53	0.64	0.46	0.63
MLP 1000 relu 0.001	0.53	0.66	0.49	0.57
SVC	0.52	0.65	0.47	0.59
Naive Bayes	0.52	0.61	0.44	0.64
MLP 300 200 100	0.52	0.65	0.47	0.59
Logistic Regression	0.52	0.62	0.44	0.64
MLP 1000 relu adam	0.52	0.66	0.48	0.57
Extra Trees	0.51	0.52	0.38	0.78
Gradient Boosting	0.51	0.45	0.36	0.87
Bagging	0.48	0.68	0.52	0.45
XGBoost	0.47	0.59	0.4	0.56
KNN	0.45	0.55	0.37	0.57
Linear SVC	0.0	0.67	0.0	0.0
SGD	0.0	0.67	0.0	0.0

Table 2.4 Detailed results for ResNet50 model on ProMapCz dataset.

2.2.4 Inception V3

Inception V3 [23] is a CNN architecture, part of the inception family, and very efficient in image analysis and object detection. Characterizing this would be the use of mixed convolutional layers and the introduction of factorized convolutions that help reduce the number of parameters, making the network much more efficient. Coupled with this are label smoothing and batch normalization, which make it robust and highly performing on many image recognition tasks.

Architecture of Inception Model

The Inception architecture uses different convolutional filters within a single layer, capturing various levels of detail. Each module's output is concatenated along the depth dimension, enabling the network to process multi-scale features simultaneously.

Inception Module

The output of an Inception module can be formulated as:

$$\text{Output} = [\text{Conv}_{1\times1}, \text{Conv}_{3\times3}, \text{Conv}_{5\times5}, \text{MaxPool}_{3\times3}]$$

where Conv represents convolutional operations with specified filter sizes.

Key Components

- **Convolutional Layers**: With various filter sizes (1×1, 3×3, 5×5).

- **Dimensionality Reduction**: 1×1 convolutions to decrease the number of input channels.
- **Pooling Layers**: Max-pooling

Results

We performed the same experiments as in the case of previous feature extraction models.

Results on ProMapEn We obtained the highest result on the ProMapEn dataset using the MLP model, using only image data. The highest F1 score was 0.62 using the following MLP 1.4. The F1 score for the ProMapEn dataset is 0.62. Detailed results for the ProMapEn dataset are as follows:

Model	F1	Accuracy	Precision	Recall
MLP 1000 tanh	0.62	0.69	0.52	0.76
Naive Bayes	0.62	0.69	0.51	0.77
Logistic Regression	0.62	0.69	0.52	0.76
SVC	0.6	0.68	0.5	0.74
XGBoost	0.58	0.64	0.46	0.78
MLP 1000 relu adam	0.57	0.6	0.44	0.81
MLP 1000	0.56	0.58	0.42	0.82
MLP 500 500	0.56	0.59	0.43	0.82
MLP 300 200 100	0.56	0.61	0.44	0.76
Random Forest	0.56	0.62	0.45	0.75
MLP 1000 relu 0.001	0.56	0.58	0.42	0.82
AdaBoost	0.56	0.58	0.42	0.82
Extra Trees	0.56	0.61	0.44	0.76
Gradient Boosting	0.53	0.5	0.38	0.86
KNN	0.51	0.59	0.41	0.65
Bagging	0.4	0.73	0.7	0.28
SGD	0.39	0.73	0.76	0.26
Linear SVC	0.38	0.74	0.81	0.25

Table 2.5 Detailed results for Inception V3 model on ProMapEn dataset (only image data).

Results on ProMapCz We performed the same experiments as in the case of previous models on Czech data. For the ProMapCz dataset, the highest F1 score achieved was 0.53 using a Random Forest classifier 1.4 with only image data. Detailed results for the ProMapCz dataset are as follows:

Model	F1	Accuracy	Precision	Recall
Random Forest	0.53	0.58	0.42	0.72
Extra Trees	0.53	0.57	0.41	0.76
MLP 1000 relu adam	0.52	0.61	0.44	0.65
MLP 500 500	0.52	0.64	0.46	0.58
XGBoost	0.51	0.62	0.44	0.59
MLP 300 200 100	0.51	0.64	0.46	0.56
MLP 1000 relu 0.001	0.51	0.65	0.47	0.57
Logistic Regression	0.51	0.6	0.42	0.65
MLP 1000 tanh	0.51	0.59	0.42	0.65
MLP 1000	0.51	0.65	0.47	0.57
Naive Bayes	0.51	0.59	0.42	0.65
AdaBoost	0.49	0.64	0.45	0.53
Gradient Boosting	0.49	0.6	0.42	0.58
SVC	0.49	0.64	0.45	0.54
KNN	0.47	0.6	0.41	0.55
Bagging	0.44	0.68	0.51	0.38
Linear SVC	0.35	0.69	0.57	0.26
SGD	0.0	0.67	0.0	0.0

Table 2.6 Detailed results for Inception V3 model on ProMapCz dataset (only image data).

2.2.5 EfficientNet

EfficientNet [24] is a method for scaling convolutional neural network architectures using compound coefficients, scaling up depth, width, and resolution. In contrast to the arbitrary scaling of depth, width, and resolutions usually done, EfficientNet uses a set of fixed coefficients to scale network width, network depth, and test resolution uniformly.

Architecture of EfficientNet

The EfficientNet family uses a baseline network called EfficientNet-B0, scaled to create a range of models from EfficientNet-B0 to EfficientNet-B7. The key innovation is the compound scaling method, which uniformly scales all depth, width, and resolution dimensions.

Key Components

- **Mobile Inverted Bottleneck Convolution (MBConv)**: EfficientNet uses MBConv layers with squeeze-and-excitation optimization.
- **Swish Activation Function**: A smooth, non-monotonic activation function.
- **Compound Scaling**: Simultaneously scales network width, depth, and resolution.

2.2.6 EfficientNet Variants

The EfficientNet family includes:

- **EfficientNet-B0**: Baseline model
- **EfficientNet-B1 to B7**: Scaled models with increasing complexity and accuracy

Reults

Results on ProMapEn The highest F1 score, 0.63, was obtained using EfficientNet in combination with MLP 1.4. Detailed results for the ProMapEn dataset are as follows:

Model	F1	Accuracy	Precision	Recall
MLP 1000 relu adam	0.63	0.74	0.58	0.69
SGD	0.62	0.78	0.71	0.55
Naive Bayes	0.61	0.69	0.51	0.74
MLP 500 500	0.6	0.67	0.49	0.75
Logistic Regression	0.6	0.68	0.51	0.74
MLP 300 200 100	0.6	0.68	0.51	0.73
MLP 1000 tanh	0.6	0.68	0.51	0.74
MLP 1000	0.59	0.67	0.49	0.74
SVC	0.59	0.68	0.51	0.7
Random Forest	0.59	0.6	0.44	0.86
MLP 1000 relu 0.001	0.59	0.67	0.49	0.74
AdaBoost	0.59	0.64	0.47	0.78
Bagging	0.58	0.78	0.77	0.47
Extra Trees	0.56	0.57	0.42	0.83
XGBoost	0.56	0.61	0.44	0.77
Linear SVC	0.55	0.78	0.81	0.42
KNN	0.54	0.53	0.39	0.83
Gradient Boosting	0.54	0.51	0.39	0.9

Table 2.7 Detailed results for EfficientNet model on ProMapEn dataset (only image data)

Results on ProMapCz For the ProMapCz dataset, we performed the same experiments. The highest F1 score achieved was 0.56 using a Random Forest classifier 1.4 with only image data.

The F1 score for the ProMapCz dataset is 0.56. Detailed results for the ProMapCz dataset are as follows:

Model	F1	Accuracy	Precision	Recall
Random Forest	0.56	0.58	0.43	0.82
SGD	0.55	0.66	0.48	0.63
MLP 1000 relu adam	0.55	0.67	0.49	0.61
Logistic Regression	0.55	0.63	0.46	0.67
MLP 500 500	0.54	0.65	0.48	0.63
MLP 1000 relu 0.001	0.54	0.65	0.47	0.63
MLP 300 200 100	0.54	0.65	0.47	0.63
MLP 1000 tanh	0.54	0.62	0.45	0.68
MLP 1000	0.54	0.65	0.47	0.63
Naive Bayes	0.54	0.61	0.44	0.68
SVC	0.53	0.64	0.46	0.63
AdaBoost	0.53	0.63	0.45	0.64
Extra Trees	0.52	0.63	0.45	0.62
XGBoost	0.51	0.61	0.43	0.61
Gradient Boosting	0.5	0.61	0.43	0.59
KNN	0.49	0.57	0.4	0.64
Bagging	0.48	0.67	0.5	0.46
Linear SVC	0.42	0.69	0.53	0.35

Table 2.8 Detailed results for EfficientNet model on ProMapCz dataset (only image data).

2.2.7 MaxOutputEnsemble Custom Model

The MaxOutputEnsemble custom model combines VGG16, ResNet50, Inception V3, and EfficientNet results. By taking the maximum value from the output of these models, the MaxOutputEnsemble ensures that the most significant features are captured for similarity computation.

Results

Results on ProMapEn The highest F1 score is 0.62 for ProMapEn (using only the images) using the SVC 1.4. Detailed results for the PromapEn dataset are as follows:

Model	F1	Accuracy	Precision	Recall
SVC	0.62	0.74	0.59	0.65
Logistic Regression	0.61	0.7	0.53	0.72
Naive Bayes	0.6	0.68	0.5	0.76
MLP 1000 relu adam	0.6	0.76	0.64	0.57
MLP 1000 tanh	0.59	0.67	0.49	0.74
Bagging	0.59	0.77	0.7	0.5
MLP 500 500	0.59	0.68	0.5	0.71
MLP 300 200 100	0.59	0.71	0.55	0.64
AdaBoost	0.59	0.68	0.5	0.73
Extra Trees	0.59	0.69	0.51	0.7
Random Forest	0.58	0.69	0.52	0.66
Linear SVC	0.57	0.77	0.72	0.47
Gradient Boosting	0.57	0.7	0.53	0.61
MLP 1000	0.55	0.61	0.44	0.71
MLP 1000 relu 0.001	0.55	0.61	0.44	0.71
XGBoost	0.52	0.62	0.44	0.63
KNN	0.52	0.62	0.44	0.64
SGD	0.09	0.68	0.5	0.05

Table 2.9 Detailed results for MaxOutputEnsemble custom model on ProMapEn dataset (only image data).

Results on ProMapCz For the ProMapCz dataset, the highest F1 score achieved was 0.56 using the XGBoost classifier 1.4 with only image data.

The F1 score for the ProMapCz dataset is 0.56. Detailed results for the ProMapCz dataset are as follows:

Model	F1	Accuracy	Precision	Recall
XGBoost	0.56	0.61	0.44	0.78
MLP 300 200 100	0.56	0.66	0.49	0.66
MLP 1000 relu 0.001	0.56	0.66	0.49	0.66
MLP 1000	0.56	0.66	0.49	0.66
Extra Trees	0.54	0.64	0.47	0.63
MLP 1000 relu adam	0.54	0.65	0.47	0.63
Logistic Regression	0.54	0.64	0.46	0.65
MLP 1000 tanh	0.54	0.63	0.45	0.66
MLP 500 500	0.54	0.65	0.48	0.62
Naive Bayes	0.54	0.65	0.47	0.63
Random Forest	0.54	0.66	0.48	0.61
SVC	0.53	0.66	0.48	0.59
AdaBoost	0.53	0.63	0.46	0.64
Gradient Boosting	0.53	0.55	0.4	0.79
KNN	0.48	0.53	0.38	0.67
Bagging	0.47	0.67	0.49	0.45
Linear SVC	0.45	0.69	0.53	0.39
SGD	0.29	0.7	0.67	0.18

Table 2.10 Detailed results for MaxOutputEnsemble custom model on ProMapCz dataset (only image data).

2.2.8 Conclusion about using only image data

We reached the best results by EfficientNet and MaxOutputEnsemble custom models. Both of them have comparatively similar performance. However, the interesting thing about MaxOutputEnsemble is that it has information about four different models, which makes it better suitable for combined models of text and images (we will see in the next section).

Model	Preprocessed Model Name	F1	Acc	Prec	Rec
efficientnet	MLP 1000 relu adam	0.63	0.74	0.58	0.69
inception	MLP 1000 tanh	0.62	0.69	0.52	0.76
MaxOutputEnsemble	SVC	0.62	0.74	0.59	0.65
inception	Naive Bayes	0.62	0.69	0.51	0.77
inception	Logistic Regression	0.62	0.69	0.52	0.76
efficientnet	SGD	0.62	0.78	0.71	0.55
efficientnet	Naive Bayes	0.61	0.69	0.51	0.74
MaxOutputEnsemble	Logistic Regression	0.61	0.7	0.53	0.72
efficientnet	MLP 500 500	0.6	0.67	0.49	0.75
resnet	MLP 1000 relu adam	0.6	0.73	0.57	0.64
efficientnet	MLP 300 200 100	0.6	0.68	0.51	0.73
inception	SVC	0.6	0.68	0.5	0.74
efficientnet	Logistic Regression	0.6	0.68	0.51	0.74
efficientnet	MLP 1000 tanh	0.6	0.68	0.51	0.74
MaxOutputEnsemble	Naive Bayes	0.6	0.68	0.5	0.76
MaxOutputEnsemble	MLP 1000 relu adam	0.6	0.76	0.64	0.57
MaxOutputEnsemble	MLP 500 500	0.59	0.68	0.5	0.71
resnet	Naive Bayes	0.59	0.7	0.53	0.66
resnet	MLP 500 500	0.59	0.68	0.51	0.7
MaxOutputEnsemble	Bagging	0.59	0.77	0.7	0.5

Table 2.11 Detailed results for MaxOutputEnsemble custom model on ProMapEn dataset (only image data)

Model	Preprocessed Model Name	F1	Acc	Prec	Rec
MaxOutputEnsemble	MLP 1000 relu 0.001	0.56	0.66	0.49	0.66
efficientnet	Random Forest	0.56	0.58	0.43	0.82
MaxOutputEnsemble	MLP 300 200 100	0.56	0.66	0.49	0.66
MaxOutputEnsemble	XGBoost	0.56	0.61	0.44	0.78
MaxOutputEnsemble	MLP 1000	0.56	0.66	0.49	0.66
efficientnet	SGD	0.55	0.66	0.48	0.63
efficientnet	Logistic Regression	0.55	0.63	0.46	0.67
efficientnet	MLP 1000 relu adam	0.55	0.67	0.49	0.61
vgg	Naive Bayes	0.54	0.64	0.46	0.66
vgg	MLP 500 500	0.54	0.67	0.5	0.59
vgg	MLP 1000 tanh	0.54	0.64	0.46	0.66
efficientnet	MLP 1000 relu 0.001	0.54	0.65	0.47	0.63
vgg	MLP 300 200 100	0.54	0.68	0.5	0.59
efficientnet	MLP 1000 tanh	0.54	0.62	0.45	0.68
efficientnet	MLP 500 500	0.54	0.65	0.48	0.63
MaxOutputEnsemble	Naive Bayes	0.54	0.65	0.47	0.63
efficientnet	MLP 1000	0.54	0.65	0.47	0.63
MaxOutputEnsemble	MLP 1000 relu adam	0.54	0.65	0.47	0.63
efficientnet	Naive Bayes	0.54	0.61	0.44	0.68
MaxOutputEnsemble	MLP 500 500	0.54	0.65	0.48	0.62

Table 2.12 Detailed results for MaxOutputEnsemble custom model on ProMapCz dataset (only image data)

3 Comparison of Text Representation Models

3.1 Introduction

Text representation plays a vital role in the performance of the Product Mapping task. We aim to compare several text representation models ranging from traditional methods like TF-IDF to advanced transformer-based architectures such as BERT, RoBERTa, and Sentence Transformers. We evaluated the models on several datasets, including ProMapEn, ProMapCz, Amazon-Walmart, and Amazon-Google.

We used the following approach to handle the data and evaluate the models. For traditional models, we focused on features derived from TF-IDF and Word2Vec, whereas for advanced models, we leveraged pretrained transformer-based architectures fine-tuned for specific NLP tasks. The uniqueness of datasets evaluates models more comprehensively.

3.1.1 Data Processing Workflow

To visualize the data processing workflow, we illustrate the two approaches used. The first one is text Processing with chosen NLP extractor methods (BERT, RoBERTa, TF-IDF, etc.). The second one calculates cosine similarities for each feature and trains the models to detect corresponding and different products.

The datasets consist of features such as *name1, name2, short_description1, short_description2, long_description1, long_description2*, and for the ProMapEn and ProMapCz datasets, additional features like *specification1* and *specification2*. Each dataset is processed using the chosen NLP extractor method (BERT, RoBERTa, TF-IDF) to extract embeddings. Cosine similarities are then calculated for each feature, resulting in a dataset with either one feature (if only name is used) or four features (cosine values for name, short description, long description, and specification). This processed data subsequently trains classifiers such as MLP, Logistic Regression, Naive Bayes, and SVC, enabling effective Product Mapping.

3.2 Traditional Models

3.2.1 TF-IDF

Overview of TF-IDF

One of the challenges in Product Matching is that there are words that are either very common or words that are rare. TF-IDf [13] technique deals with this problem. We experimented with TF-IDF on ProMapEn, ProMapCz, Amazon-Walmart, and Amazon-Google datasets. The experiments are done by using only name features and all textual features. TF-IDF stands for Term Frequency-Inverse Document Frequency. This measure of the importance of words in a document with respect to a corpus uses two metrics: term frequency, which quantifies

the number of times that a term appears in a document, and inverse document frequency, which measures the importance of the term. This will be helpful when filtering less critical words from those most essential in the document.

Results

We used ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart for our experiments to find the best model for the Product Mapping task using text features.

ProMapEn The highest F1 score is 0.57 using all text features and is achieved by MLP 1.4,

Model	F1	Acc	Prec	Rec	Name
SVC	0.59	0.64	0.47	0.78	True
Logistic Regression	0.57	0.62	0.45	0.78	False
MLP 1000 relu adam	0.57	0.57	0.42	0.86	True
Naive Bayes	0.57	0.6	0.44	0.82	True
MLP 1000 tanh	0.57	0.59	0.43	0.83	False
Logistic Regression	0.57	0.6	0.44	0.82	True
MLP 1000 relu 0.001	0.56	0.57	0.42	0.85	True
MLP 1000 tanh	0.56	0.57	0.42	0.85	True
MLP 1000	0.56	0.57	0.42	0.85	True
MLP 1000 relu adam	0.56	0.58	0.42	0.84	False
SVC	0.55	0.57	0.42	0.8	False
Naive Bayes	0.55	0.58	0.42	0.78	False
MLP 300 200 100	0.55	0.54	0.4	0.87	True
AdaBoost	0.55	0.54	0.4	0.88	False
Bagging	0.54	0.7	0.54	0.54	False
Extra Trees	0.54	0.54	0.4	0.82	False
Random Forest	0.54	0.53	0.4	0.86	False
Random Forest	0.53	0.48	0.37	0.88	True
MLP 500 500	0.53	0.49	0.38	0.89	True
XGBoost	0.53	0.51	0.38	0.85	False

Table 3.1 Results for ProMapEn using TF-IDF (only text data). The Name column shows if only "name" features were used for comparison or all text features.

We found out that in most cases, the name feature has similar results as using all features and sometimes even outperforms. The highest F1 score achieved is 0.59 using the name feature, which is greater than the F1 score achieved by using all features. The model used is Linear Support Vector Classification 1.4.

ProMapCz The highest F1 score is 0.55 using all text features and is achieved by MLP classifier 1.4

Model	F1	Acc	Prec	Rec	Name
MLP 1000	0.55	0.57	0.42	0.82	False
Random Forest	0.55	0.6	0.44	0.73	False
MLP 1000 relu 0.001	0.55	0.57	0.42	0.82	False
MLP 1000 tanh	0.54	0.55	0.4	0.8	True
Logistic Regression	0.54	0.55	0.4	0.8	True
MLP 1000 relu adam	0.54	0.56	0.41	0.78	True
XGBoost	0.53	0.5	0.38	0.85	False
MLP 300 200 100	0.53	0.47	0.37	0.91	False
MLP 1000 relu 0.001	0.53	0.57	0.41	0.73	True
MLP 1000	0.53	0.57	0.41	0.73	True
Naive Bayes	0.53	0.53	0.4	0.83	True
AdaBoost	0.53	0.43	0.36	0.96	False
Gradient Boosting	0.53	0.51	0.39	0.83	False
Extra Trees	0.53	0.43	0.36	0.97	False
MLP 300 200 100	0.52	0.51	0.38	0.82	True
MLP 500 500	0.52	0.5	0.38	0.83	True
AdaBoost	0.52	0.57	0.41	0.72	True
MLP 1000 relu adam	0.52	0.44	0.36	0.94	False
MLP 500 500	0.52	0.46	0.37	0.9	False
KNN	0.52	0.59	0.42	0.69	False

Table 3.2 Results for ProMapCz using TF-IDF (only text data).

The highest F1 result using only the name feature is 0.54 using Logistic Regression. TF-IDF's advantage in Product Matching is that it easily adapts to new domains. However, one of the challenges of Product Mapping is that it has many small sub-domains.

Amazon-Walmart The highest F1 score achieved is 0.82 using all text features and is achieved by MLP classifier 1.4. Interestingly, MLP classifiers with "tanh" activation are very effective in our task.

Model	F1	Acc	Prec	Rec	Name
Bagging	0.82	0.86	0.8	0.83	False
SVC	0.82	0.86	0.75	0.9	False
Extra Trees	0.82	0.86	0.77	0.87	False
MLP 1000 tanh	0.82	0.85	0.74	0.91	False
MLP 1000 relu adam	0.82	0.86	0.81	0.82	False
AdaBoost	0.81	0.85	0.75	0.89	False
Logistic Regression	0.81	0.85	0.74	0.89	False
MLP 1000 relu 0.001	0.8	0.84	0.74	0.86	True
MLP 500 500	0.8	0.84	0.75	0.86	True
MLP 1000	0.8	0.84	0.74	0.86	True
SGD	0.8	0.84	0.75	0.85	False
AdaBoost	0.8	0.84	0.74	0.86	True
Naive Bayes	0.8	0.84	0.74	0.88	False
MLP 1000 relu 0.001	0.79	0.84	0.78	0.8	False
Bagging	0.79	0.84	0.75	0.84	True
MLP 1000 relu adam	0.79	0.84	0.76	0.82	True
MLP 1000 tanh	0.79	0.83	0.73	0.86	True
MLP 300 200 100	0.79	0.83	0.73	0.86	True
Naive Bayes	0.79	0.83	0.73	0.86	True
SVC	0.79	0.83	0.72	0.86	True

Table 3.3 Results for Amazon-Walmart using TF-IDF (only text data).

We have an F1 result of 0.8 using MLP classifiers for only using the name feature.

Amazon-Google The highest F1 score achieved is 0.8 using all text features and is achieved by MLP classifier 1.4

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu 0.001	0.8	0.85	0.9	0.72	False
MLP 300 200 100	0.8	0.84	0.79	0.81	False
Extra Trees	0.8	0.84	0.81	0.8	False
MLP 1000 relu adam	0.8	0.84	0.81	0.78	False
MLP 1000	0.8	0.85	0.9	0.72	False
MLP 500 500	0.8	0.83	0.76	0.84	False
Bagging	0.79	0.85	0.89	0.71	False
AdaBoost	0.79	0.84	0.87	0.72	False
XGBoost	0.79	0.83	0.78	0.8	False
Random Forest	0.79	0.83	0.79	0.79	False
KNN	0.78	0.82	0.76	0.81	False
SVC	0.78	0.82	0.76	0.79	False
Extra Trees	0.77	0.78	0.67	0.91	True
Bagging	0.77	0.78	0.67	0.9	True
Naive Bayes	0.77	0.79	0.68	0.9	True
AdaBoost	0.77	0.78	0.67	0.9	True
MLP 1000 relu 0.001	0.77	0.78	0.67	0.9	True
MLP 1000 relu adam	0.77	0.78	0.67	0.9	True
MLP 1000 tanh	0.77	0.78	0.67	0.9	True
MLP 300 200 100	0.77	0.78	0.67	0.91	True

Table 3.4 Results for Amazon-Google using TF-IDF (only text data).

Using only the name feature is outperformed by using all features for this dataset.

3.2.2 Word2Vec

Overview of Word2Vec

Word2Vec [25] Word2Vec is the representation of words as vectors in high-dimensional space—a technique in natural language processing. The vectors capture semantic meaning based on the context in which they are used. Some shallow neural network architecture will train the word2vec model on huge text corpora to learn word associations from the surrounding text.

Formalizing these two critical aspects of word2vec, there are basically two ways to implement it: CBOW: The model is going to predict the current word from its context. It creates an average of the vectors of the context words and uses this as a prediction for the current word. Skip-gram: The model predicts the context words based on the current word. In this case, each pair of contexts is treated as a new observation. Both models rely on a method called negative sampling, in which the model is trained to distinguish a target word against random noise samples. Thus, this improves the quality and speed of training.

List of Word2Vec Models

We used the following models in our experiments for getting the embeddings of text data. After getting the embeddings, we used cosine similarity to find the

similarity distance between corresponding features (e.g., "name1" vs. "name2", "short_description1" vs. "short_description2"). Some of them proved to be better than others ones.

- **fasttext-wiki-news-subwords-300**
- **conceptnet-numberbatch-17-06-300**
- **word2vec-ruscorpora-300**
- **word2vec-google-news-300**
- **glove-wiki-gigaword-50**
- **glove-wiki-gigaword-100**
- **glove-wiki-gigaword-200**
- **glove-wiki-gigaword-300**
- **glove-twitter-25**
- **glove-twitter-50**
- **glove-twitter-100**
- **glove-twitter-200**
- **___testing__word2vec-matrix-synopsis**

Results and Explanation

We tried the models on ProMapEn, ProMapCz and Amazon-Google datasets. We do not include the results of Amazon-Walmart as it was not competitive with the results of using other text preprocessing models and didn't give us insights about the datasets and what can be improved.

Results on ProMapEn

We trained models using only the name feature and all features. Particularly, "glove-wiki-gigaword-300" and "word2vec-google-news-30" showed higher results. We can see that using only the name attribute with 'word2vec-google-news-300', we get similar results when using all features. Interestingly, using only names gives higher recall, meaning that more positive results are predicted correctly. The best result that has the highest accuracy and F1 score is done by using Logistic Regression 1.4

Model	Name	F1	Prec	Rec	Acc	Classifier
wiki-gigaword-300	False	0.61	0.58	0.63	0.73	Logistic Regression
wiki-gigaword-300	False	0.61	0.56	0.66	0.72	MLP 1000 tanh
google-news-300	True	0.61	0.49	0.8	0.67	Naive Bayes
google-news-300	True	0.61	0.49	0.8	0.67	MLP 1000 tanh
google-news-300	True	0.61	0.49	0.8	0.67	MLP 1000 relu adam
google-news-300	False	0.61	0.49	0.79	0.67	MLP 1000 relu adam
wiki-gigaword-300	True	0.6	0.58	0.62	0.73	Logistic Regression
wiki-gigaword-300	True	0.6	0.58	0.62	0.73	Naive Bayes
wiki-gigaword-300	True	0.6	0.58	0.62	0.73	MLP 1000 relu adam
google-news-300	True	0.6	0.5	0.77	0.67	Logistic Regression
google-news-300	False	0.6	0.47	0.81	0.64	MLP 1000 tanh
wiki-gigaword-300	True	0.59	0.58	0.6	0.73	MLP 1000 tanh
wiki-gigaword-300	True	0.59	0.53	0.66	0.7	AdaBoost
wiki-gigaword-200	False	0.59	0.57	0.61	0.73	MLP 1000 relu adam
google-news-300	True	0.59	0.51	0.69	0.68	MLP 500 500
google-news-300	True	0.59	0.51	0.69	0.68	AdaBoost
wiki-gigaword-200	True	0.59	0.53	0.65	0.7	Logistic Regression
wiki-gigaword-200	True	0.59	0.54	0.65	0.7	MLP 1000
wiki-gigaword-200	True	0.59	0.54	0.65	0.71	MLP 500 500
wiki-gigaword-200	True	0.59	0.53	0.66	0.7	MLP 300 200 100
wiki-gigaword-200	True	0.59	0.54	0.65	0.7	MLP 1000 relu 0.001
wiki-gigaword-200	True	0.59	0.52	0.67	0.69	AdaBoost
google-news-300	False	0.59	0.47	0.81	0.64	Logistic Regression
google-news-300	False	0.59	0.48	0.76	0.65	Naive Bayes
wiki-gigaword-200	False	0.58	0.5	0.68	0.68	MLP 1000 tanh

Table 3.5 Top 25 results for ProMapEn using Word2Vec models (only text data).

Reuslts on ProMapCz

Like ProMapEn, we trained the models by using only the name and all text features. Compared to ProMapEn, the results are lower. Using only the name attribute proves to be more effective. We can assume that glove-wiki-gigaword-*** models better understand the Czech language than other Word2Vec models. The highest result is achieved by using MLP architecture 1.4

Model	Name	F1	Prec	Rec	Acc	Classifier
wiki-gigaword-200	True	0.53	0.38	0.89	0.49	MLP 300 200 100
wiki-gigaword-200	True	0.52	0.35	0.99	0.39	Random Forest
wiki-gigaword-200	True	0.52	0.37	0.88	0.46	MLP 1000
wiki-gigaword-200	True	0.52	0.36	0.95	0.42	MLP 500 500
wiki-gigaword-200	True	0.52	0.37	0.88	0.46	MLP 1000 relu 0.001
wiki-gigaword-200	True	0.52	0.35	0.98	0.4	Extra Trees
google-news-300	False	0.52	0.36	0.93	0.43	KNN
twitter-100	False	0.52	0.35	0.98	0.4	SVC
wiki-gigaword-100	False	0.52	0.39	0.8	0.53	MLP 300 200 100
wiki-gigaword-300	True	0.52	0.35	0.99	0.39	MLP 1000
wiki-gigaword-300	True	0.52	0.35	0.97	0.41	MLP 500 500
wiki-gigaword-300	True	0.52	0.35	0.95	0.42	MLP 300 200 100
wiki-gigaword-300	True	0.52	0.35	0.99	0.39	MLP 1000 relu 0.001
wiki-gigaword-50	True	0.52	0.37	0.84	0.48	SVC
twitter-200	False	0.52	0.37	0.9	0.46	MLP 1000
twitter-200	False	0.52	0.37	0.9	0.46	MLP 1000 relu 0.001
twitter-50	True	0.51	0.34	0.96	0.38	MLP 500 500
twitter-50	False	0.51	0.34	0.99	0.36	Random Forest
twitter-50	False	0.51	0.36	0.89	0.44	KNN
wiki-gigaword-200	True	0.51	0.34	0.97	0.38	Logistic Regression
wiki-gigaword-200	True	0.51	0.35	0.92	0.42	SVC
wiki-gigaword-200	True	0.51	0.34	0.97	0.38	Naive Bayes
wiki-gigaword-200	True	0.51	0.34	0.97	0.38	MLP 1000 tanh
wiki-gigaword-200	True	0.51	0.35	0.98	0.39	Gradient Boosting
twitter-100	True	0.51	0.34	0.99	0.36	Logistic Regression

Table 3.6 Top 25 results for ProMapCz using Word2Vec models (only text data).

Results on Amazon-Google

word2vec-google-news-300 proves to be the best model compared to other models, both using the name feature and using all of the features. Similar results can mean that the name feature alone has important information about the product, and a long description has lots of repetitive information in most of the products. The best F1 score is 0.92 is achieved by MLP architecture 1.4

Model	Name	F1	Prec	Rec	Acc	Classifier
google-news-300	False	0.92	0.93	0.92	0.94	MLP 500 500
google-news-300	False	0.92	0.93	0.91	0.94	Extra Trees
google-news-300	True	0.91	0.92	0.89	0.93	Logistic Regression
google-news-300	True	0.91	0.93	0.89	0.93	SVC
google-news-300	True	0.91	0.92	0.89	0.93	Naive Bayes
google-news-300	True	0.91	0.93	0.89	0.93	MLP 1000
google-news-300	True	0.91	0.92	0.89	0.92	MLP 500 500
google-news-300	True	0.91	0.93	0.89	0.93	MLP 1000 tanh
google-news-300	True	0.91	0.93	0.89	0.93	MLP 1000 relu 0.001
google-news-300	True	0.91	0.93	0.89	0.93	AdaBoost
google-news-300	True	0.91	0.94	0.88	0.93	Extra Trees
google-news-300	True	0.91	0.93	0.89	0.93	Bagging
google-news-300	True	0.91	0.93	0.89	0.93	Linear SVC
google-news-300	False	0.91	0.94	0.88	0.93	Logistic Regression
google-news-300	False	0.91	0.94	0.89	0.93	Naive Bayes
google-news-300	False	0.91	0.94	0.88	0.93	MLP 1000
google-news-300	False	0.91	0.94	0.89	0.93	MLP 300 200 100
google-news-300	False	0.91	0.94	0.88	0.93	MLP 1000 tanh
google-news-300	False	0.91	0.94	0.89	0.93	MLP 1000 relu adam
google-news-300	False	0.91	0.94	0.88	0.93	MLP 1000 relu 0.001
google-news-300	False	0.91	0.94	0.89	0.93	AdaBoost
google-news-300	False	0.91	0.93	0.89	0.93	Bagging
google-news-300	False	0.91	0.94	0.88	0.93	Linear SVC
google-news-300	False	0.91	0.93	0.88	0.93	SGD
google-news-300	True	0.9	0.9	0.89	0.92	MLP 300 200 100

Table 3.7 Top 25 results for Amazon-Google using Word2Vec models (only text data).

3.3 Transformer-Based Models

3.3.1 BERT Models

BERT [7] architecture brings advancements in NLP applications. One of the advantages of transformer architecture is that it allows the model to track the position of the words in a sequence, which we find beneficial in the Product Mapping task. [26] We will see how the models based on BERT architecture have advantage towards traditional NLP methods.

BERT-base-uncased

BERT-base-uncased is one of the versions of the BERT with the following characteristics

- **Layers**: 12
- **Hidden Units**: 768
- **Attention Heads**: 12

- **Parameters**: 110 million
- **uncased**: no difference between lowercase and uppercase

When we compare cased and uncased models, we can see that the case is important in the Product Matching task.

Results and Explanation

We tried the models on ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart.

Results on ProMapEn

The highest F1 result that we reached using all text features is 0.55 with MLP model architecture 1.4 The results are worse than those of the previous techniques, which proves the case's importance in our task.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 tanh	0.55	0.68	0.5	0.6	False
Logistic Regression	0.53	0.61	0.44	0.67	False
MLP 1000 relu adam	0.53	0.5	0.38	0.87	False
AdaBoost	0.52	0.69	0.53	0.51	False
Naive Bayes	0.52	0.57	0.41	0.7	False
MLP 300 200 100	0.52	0.49	0.37	0.84	False
MLP 500 500	0.51	0.53	0.39	0.74	False
Random Forest	0.5	0.59	0.42	0.63	False
SVC	0.5	0.36	0.33	0.99	False
Extra Trees	0.49	0.39	0.34	0.91	False
Gradient Boosting	0.49	0.32	0.32	1.0	False
XGBoost	0.49	0.32	0.32	1.0	False
MLP 1000 relu 0.001	0.49	0.43	0.35	0.85	False
MLP 1000	0.49	0.43	0.35	0.85	False
KNN	0.47	0.41	0.33	0.81	False
Bagging	0.25	0.63	0.36	0.19	False
Linear SVC	0.0	0.68	0.0	0.0	False
SGD	0.0	0.68	0.0	0.0	False

Table 3.8 Results for ProMapEn dataset using bert-base-uncased (only text data).

ProMapCz dataset

The highest F1 result we reached using all text features is 0.52 with MLP model 1.4

Again, the results prove to be less than those of previous techniques. The brand names are treated as regular words (e.g., Apple = apple).

Model	F1	Acc	Prec	Rec	Name
MLP 1000	0.53	0.49	0.38	0.88	True
MLP 500 500	0.53	0.5	0.38	0.88	True
MLP 1000 relu adam	0.53	0.49	0.38	0.88	True
MLP 1000 relu 0.001	0.53	0.49	0.38	0.88	True
Logistic Regression	0.53	0.49	0.38	0.88	True
MLP 300 200 100	0.53	0.47	0.37	0.89	True
Naive Bayes	0.53	0.49	0.38	0.88	True
MLP 300 200 100	0.52	0.42	0.36	0.94	False
AdaBoost	0.52	0.44	0.36	0.92	True
MLP 1000 tanh	0.52	0.49	0.38	0.86	True
Naive Bayes	0.51	0.39	0.35	0.95	False
MLP 1000	0.51	0.4	0.35	0.97	False
MLP 500 500	0.51	0.4	0.35	0.97	False
MLP 1000 tanh	0.51	0.42	0.35	0.92	False
Random Forest	0.51	0.43	0.36	0.91	True
MLP 1000 relu 0.001	0.51	0.4	0.35	0.97	False
Extra Trees	0.51	0.57	0.41	0.69	True
Logistic Regression	0.51	0.37	0.34	0.99	False
MLP 1000 relu adam	0.51	0.43	0.36	0.92	False
SVC	0.5	0.34	0.33	1.0	True

Table 3.9 Results for ProMapCz dataset using bert-base-uncased (only text data).

The results of using only the name feature are higher. The important observation is that Naive Bayes outperformed MLP models by being faster and using fewer resources. The highest F1 score for the name feature is 0.53.

Amazon-Walmart dataset

We got the same results for both the name feature and all text features. The highest F1 result is achieved by the AdaBoost 1.4.

- **n_estimators**: 1000
- **random_state**: 0
- **learning_rate**: 0.01

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu 0.001	0.76	0.82	0.74	0.79	False
SVC	0.76	0.82	0.74	0.78	False
AdaBoost	0.76	0.82	0.74	0.77	True
MLP 1000	0.76	0.82	0.74	0.79	False
AdaBoost	0.76	0.83	0.75	0.78	False
Linear SVC	0.75	0.82	0.76	0.74	False
Naive Bayes	0.75	0.81	0.73	0.77	True
MLP 1000	0.75	0.82	0.74	0.77	True
MLP 500 500	0.75	0.82	0.74	0.77	True
Extra Trees	0.75	0.81	0.71	0.8	False
MLP 300 200 100	0.75	0.82	0.74	0.77	True
MLP 1000 tanh	0.75	0.81	0.73	0.77	True
SVC	0.75	0.82	0.74	0.77	True
MLP 1000 tanh	0.75	0.81	0.72	0.79	False
MLP 300 200 100	0.75	0.81	0.73	0.77	False
MLP 500 500	0.75	0.81	0.71	0.8	False
MLP 1000 relu 0.001	0.75	0.82	0.74	0.77	True
Random Forest	0.75	0.82	0.75	0.75	False
Logistic Regression	0.75	0.81	0.73	0.77	True
Linear SVC	0.74	0.82	0.75	0.73	True

Table 3.10 Results for Amazon-Walmart dataset using bert-base-uncased (only text data).

Amazon-Google dataset

The highest F1 result achieved is 0.84 with MLP 1.4. The models trained by only the name feature have worse results, and it means that other attributes have valuable information that the name doesn't provide.

Model	F1	Acc	Prec	Rec	Name
MLP 1000	0.84	0.87	0.85	0.82	False
MLP 300 200 100	0.84	0.87	0.84	0.84	False
MLP 1000 relu 0.001	0.84	0.87	0.85	0.82	False
Bagging	0.84	0.87	0.85	0.83	False
AdaBoost	0.83	0.87	0.86	0.8	False
Extra Trees	0.83	0.86	0.83	0.83	False
SVC	0.83	0.87	0.83	0.84	False
Naive Bayes	0.83	0.86	0.83	0.82	False
MLP 500 500	0.83	0.87	0.84	0.82	False
MLP 1000 relu adam	0.83	0.86	0.83	0.84	False
Random Forest	0.82	0.86	0.83	0.82	False
Logistic Regression	0.81	0.84	0.79	0.84	False
SGD	0.81	0.84	0.79	0.84	False
Linear SVC	0.81	0.85	0.81	0.82	False
XGBoost	0.81	0.85	0.83	0.79	False
MLP 1000 tanh	0.81	0.85	0.79	0.84	False
KNN	0.81	0.85	0.83	0.79	False
Linear SVC	0.8	0.84	0.8	0.8	True
MLP 1000	0.8	0.83	0.79	0.81	True
Bagging	0.8	0.84	0.81	0.78	True

Table 3.11 Results for Amazon-Google dataset using bert-base-uncased (only text data).

BERT-large-uncased

BERT-large-uncased is one of the versions of the BERT with the following characteristics

- **Layers**: 24
- **Hidden Units**: 1024
- **Attention Heads**: 16
- **Parameters**: 340 million
- **uncased**: no difference between lowercase and uppercase

Results and Explanation

We tried the models on ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart datasets.

ProMapEn dataset

The highest F1 result that we reached using all text features is 0.54 with MLP model 1.4

The best result is less than the result in bert-base-uncased. This is because it becomes more challenging to concentrate on small domains with large contexts.

We can see that the results are worse than those of the previous techniques. That proves the case importance in our task.

Model	F1	Acc	Prec	Rec	Name
Logistic Regression	0.56	0.59	0.43	0.8	True
Naive Bayes	0.56	0.59	0.43	0.8	True
SVC	0.55	0.63	0.46	0.68	True
MLP 1000 tanh	0.55	0.58	0.42	0.8	True
MLP 300 200 100	0.55	0.59	0.43	0.75	True
MLP 500 500	0.54	0.63	0.45	0.67	True
MLP 1000 tanh	0.54	0.54	0.4	0.81	False
Logistic Regression	0.53	0.55	0.4	0.77	False
Naive Bayes	0.52	0.54	0.4	0.77	False
KNN	0.51	0.54	0.39	0.72	True
MLP 1000	0.51	0.45	0.36	0.9	True
MLP 300 200 100	0.51	0.44	0.36	0.89	False
MLP 1000 relu adam	0.51	0.38	0.34	0.98	False
MLP 500 500	0.51	0.41	0.35	0.93	False
MLP 1000 relu 0.001	0.51	0.45	0.36	0.9	True
Extra Trees	0.5	0.37	0.34	0.98	True
XGBoost	0.5	0.39	0.34	0.92	False
XGBoost	0.5	0.38	0.34	0.96	True
Gradient Boosting	0.5	0.43	0.35	0.86	False
MLP 1000 relu 0.001	0.5	0.36	0.34	0.99	False

Table 3.12 Results for ProMapEn dataset using bert-large-uncased (only text data).

However, using only name feature with bert-large-uncased and Naive Bayes gives the highest F1 score (0.56) out of all previously used BERT methods.

ProMapCz dataset

The highest F1 result that we reached using all text features is 0.52 with the Random Forest classifier. Again, the best result is less than the result in bert-base-uncased.

Model	F1	Acc	Prec	Rec	Name
Random Forest	0.52	0.45	0.36	0.9	False
XGBoost	0.51	0.48	0.37	0.81	False
Naive Bayes	0.51	0.4	0.35	0.94	True
MLP 1000 tanh	0.51	0.39	0.34	0.95	False
MLP 300 200 100	0.51	0.38	0.34	0.98	False
Logistic Regression	0.51	0.38	0.34	0.96	False
Logistic Regression	0.5	0.39	0.34	0.94	True
Extra Trees	0.5	0.34	0.33	1.0	False
AdaBoost	0.5	0.42	0.35	0.88	False
MLP 1000 relu 0.001	0.5	0.35	0.33	1.0	False
MLP 1000 relu adam	0.5	0.45	0.36	0.84	False
MLP 500 500	0.5	0.34	0.33	1.0	False
MLP 1000	0.5	0.35	0.33	1.0	False
Naive Bayes	0.5	0.39	0.34	0.93	False
SVC	0.5	0.33	0.33	1.0	False
XGBoost	0.5	0.4	0.34	0.92	True
MLP 300 200 100	0.5	0.35	0.33	0.99	True
MLP 1000 relu 0.001	0.5	0.34	0.33	0.99	True
MLP 1000	0.5	0.34	0.33	0.99	True
MLP 500 500	0.5	0.34	0.33	0.99	True

Table 3.13 Results for ProMapEn dataset using bert-large-uncased (only text data).

Amazon-Walmart dataset

AdaBoost model 1.4 with the following parameters has the highest F1 score 0.72. It is less than in bert-base-uncased (0.76).

Again, the best result is less than the result in bert-base-uncased.

Model	F1	Acc	Prec	Rec	Name
AdaBoost	0.72	0.78	0.67	0.77	False
SVC	0.72	0.77	0.65	0.79	False
Logistic Regression	0.71	0.77	0.65	0.79	False
Extra Trees	0.71	0.78	0.67	0.76	False
MLP 1000 tanh	0.71	0.77	0.64	0.8	False
MLP 300 200 100	0.7	0.76	0.63	0.78	False
Logistic Regression	0.7	0.77	0.66	0.75	True
SGD	0.7	0.76	0.66	0.74	True
MLP 500 500	0.7	0.76	0.64	0.78	False
Naive Bayes	0.7	0.77	0.65	0.76	False
Naive Bayes	0.69	0.77	0.67	0.71	True
MLP 1000 relu 0.001	0.69	0.75	0.63	0.77	False
SVC	0.69	0.77	0.67	0.7	True
MLP 1000	0.69	0.75	0.63	0.77	False
MLP 1000 tanh	0.68	0.76	0.67	0.69	True
MLP 300 200 100	0.68	0.76	0.67	0.7	True
Random Forest	0.68	0.75	0.64	0.73	False
MLP 1000 relu 0.001	0.68	0.76	0.67	0.7	True
MLP 500 500	0.68	0.76	0.66	0.71	True
MLP 1000	0.68	0.76	0.67	0.7	True

Table 3.14 Results for Amazon-Walmart dataset using bert-large-uncased (only text data).

Amazon-Google dataset

The results for bert-large-uncased and bert-base-uncased are the same for this dataset. It is 0.84, and in this case, it is reached by MLP classifier 1.4.

Again, the best result is less than the result in bert-base-uncased.

Model	F1	Acc	Prec	Rec	Name
MLP 300 200 100	0.84	0.87	0.87	0.81	False
MLP 1000 relu adam	0.84	0.87	0.82	0.85	False
MLP 1000 relu 0.001	0.83	0.86	0.82	0.84	False
Random Forest	0.83	0.87	0.84	0.83	False
Extra Trees	0.83	0.86	0.83	0.83	False
MLP 1000	0.83	0.86	0.82	0.84	False
MLP 500 500	0.83	0.87	0.84	0.83	False
AdaBoost	0.82	0.85	0.81	0.83	False
KNN	0.81	0.84	0.77	0.85	False
Gradient Boosting	0.81	0.85	0.85	0.77	False
XGBoost	0.8	0.84	0.81	0.8	False
Bagging	0.79	0.84	0.83	0.75	False
SVC	0.78	0.84	0.87	0.7	True
Naive Bayes	0.78	0.84	0.88	0.69	True
SVC	0.78	0.83	0.79	0.78	False
MLP 500 500	0.78	0.84	0.88	0.69	True
Naive Bayes	0.78	0.83	0.8	0.76	False
Bagging	0.78	0.84	0.87	0.7	True
MLP 1000 relu 0.001	0.77	0.83	0.86	0.7	True
MLP 1000 relu adam	0.77	0.83	0.86	0.7	True

Table 3.15 Results for Amazon-Google dataset using bert-large-uncased (only text data).

3.3.2 RoBERTa Models

RoBERTa has the same architecture as BERT but is trained on a larger corpus.

RoBERTa-base

Roberta-base has a similar architecture as bert-base-uncased, but it was trained on a larger corpus and is not uncased (e.g., Apple and Apple are different).

- **Layers**: 12
- **Hidden Units**: 728 in each layer
- **Attention Heads**: 16
- **Parameters**: 340 million
- **uncased**: no difference between lowercase and uppercase

Results and Explanation

We tried the models on ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart datasets.

Results on ProMapEn

The highest F1 result that we reached using all text features is 0.53 with an MLP model 1.4. Bert-large-uncased and Bert-base-uncased have higher results (0.56, 0.55) than Roberta for this dataset.

Model	F1	Acc	Prec	Rec	Name
MLP 500 500	0.53	0.45	0.37	0.93	False
MLP 1000 relu adam	0.53	0.61	0.44	0.68	False
MLP 1000	0.52	0.56	0.4	0.74	False
MLP 300 200 100	0.52	0.44	0.36	0.92	False
MLP 1000 relu 0.001	0.52	0.56	0.4	0.74	False
MLP 1000 tanh	0.5	0.41	0.35	0.9	False
Extra Trees	0.5	0.35	0.33	1.0	False
SVC	0.5	0.38	0.34	0.96	False
KNN	0.5	0.42	0.34	0.88	False
Naive Bayes	0.5	0.47	0.36	0.81	False
AdaBoost	0.49	0.32	0.32	1.0	False
Gradient Boosting	0.49	0.32	0.32	1.0	False
XGBoost	0.49	0.32	0.32	1.0	False
Logistic Regression	0.49	0.32	0.32	1.0	False
Random Forest	0.49	0.33	0.33	1.0	False
Bagging	0.07	0.66	0.33	0.04	False
Linear SVC	0.0	0.68	0.0	0.0	False
SGD	0.0	0.68	0.0	0.0	False

Table 3.16 Results for ProMapEn dataset using roberta-base (only text data).

Result on ProMapCz

Naive Bayes 1.4 has the highest result with only the name feature. Again, bert-base-uncased and bert-large-uncased performed better than roberta-base (with F1 scores 0.52 and 0.53)

Model	F1	Acc	Prec	Rec	Name
Naive Bayes	0.51	0.42	0.35	0.92	True
MLP 500 500	0.51	0.42	0.35	0.92	True
Naive Bayes	0.51	0.43	0.36	0.89	False
XGBoost	0.5	0.36	0.34	0.97	True
MLP 1000 tanh	0.5	0.42	0.35	0.89	False
Logistic Regression	0.5	0.34	0.33	1.0	True
MLP 300 200 100	0.5	0.33	0.33	1.0	True
Extra Trees	0.5	0.33	0.33	1.0	False
MLP 1000 relu 0.001	0.5	0.38	0.34	0.94	False
Random Forest	0.5	0.37	0.34	0.97	True
MLP 300 200 100	0.5	0.42	0.35	0.9	False
MLP 500 500	0.5	0.37	0.34	0.95	False
MLP 1000	0.5	0.38	0.34	0.94	False
Extra Trees	0.5	0.37	0.34	0.98	True
SVC	0.49	0.33	0.33	1.0	True
MLP 1000 relu adam	0.49	0.33	0.33	1.0	True
MLP 1000 tanh	0.49	0.33	0.33	1.0	True
AdaBoost	0.49	0.33	0.33	1.0	True
MLP 1000	0.49	0.33	0.33	1.0	True
Gradient Boosting	0.49	0.33	0.33	1.0	True

Table 3.17 Results for ProMapCz dataset using roberta-base (only text data).

Amazon-Walmart dataset

We achieve the highest result in bert-based models (bert-base-uncased, bert-large-uncased, and roberta-base) with roberta-large on the Amazon-Walmart dataset. We get an F1 score of 0.8 with the AdaBoost classifier 1.4.

Roberta-base outperforms bert-base-uncased (F1 score of 0.72) but is still less than bert-large-uncased (F1 score of 0.76)

Model	F1	Acc	Prec	Rec	Name
AdaBoost	0.74	0.79	0.68	0.81	False
MLP 1000	0.73	0.79	0.68	0.78	True
MLP 1000 relu 0.001	0.73	0.79	0.68	0.78	True
AdaBoost	0.73	0.78	0.67	0.79	True
MLP 1000 relu adam	0.73	0.81	0.74	0.71	True
Extra Trees	0.72	0.8	0.73	0.71	False
MLP 300 200 100	0.72	0.78	0.66	0.79	True
MLP 500 500	0.72	0.8	0.72	0.72	True
Naive Bayes	0.72	0.79	0.71	0.73	True
SVC	0.72	0.79	0.71	0.73	True
Logistic Regression	0.72	0.77	0.65	0.8	True
Extra Trees	0.72	0.79	0.7	0.73	True
MLP 1000 tanh	0.72	0.79	0.7	0.74	True
MLP 1000 tanh	0.72	0.8	0.72	0.73	False
MLP 300 200 100	0.72	0.78	0.67	0.79	False
MLP 500 500	0.72	0.79	0.7	0.75	False
Naive Bayes	0.72	0.79	0.71	0.74	False
MLP 1000 relu 0.001	0.71	0.79	0.72	0.7	False
Bagging	0.71	0.8	0.75	0.68	False
MLP 1000	0.71	0.79	0.72	0.7	False

Table 3.18 Results for Amazon-Walmart dataset using roberta-base (only text data).

Results on Amazon-Google

All text features are useful in the Amazon-Google dataset as they outperform models trained by only name features. We got an F1 score of 0.8 with Extra trees 1.4.

Roberta-base is under performed by both Bert models for this dataset (F1 score 0.84)

Model	F1	Acc	Prec	Rec	Name
Extra Trees	0.8	0.85	0.85	0.76	False
MLP 1000	0.8	0.84	0.83	0.78	False
MLP 300 200 100	0.8	0.85	0.86	0.75	False
MLP 1000 relu 0.001	0.8	0.84	0.83	0.78	False
Random Forest	0.79	0.85	0.89	0.71	False
AdaBoost	0.79	0.84	0.86	0.73	False
Gradient Boosting	0.78	0.85	0.92	0.68	False
MLP 300 200 100	0.78	0.83	0.82	0.74	True
MLP 500 500	0.78	0.83	0.82	0.74	True
MLP 1000	0.78	0.83	0.82	0.74	True
Naive Bayes	0.78	0.83	0.82	0.74	True
MLP 1000 relu 0.001	0.78	0.83	0.82	0.74	True
SVC	0.78	0.83	0.82	0.74	True
MLP 1000 tanh	0.78	0.83	0.82	0.74	True
MLP 1000 tanh	0.78	0.83	0.82	0.75	False
MLP 500 500	0.78	0.83	0.82	0.74	False
Naive Bayes	0.78	0.83	0.81	0.76	False
SVC	0.78	0.83	0.8	0.76	False
MLP 1000 relu adam	0.77	0.82	0.82	0.73	True
AdaBoost	0.77	0.82	0.79	0.75	True

Table 3.19 Results for Amazon-Google dataset using roberta-base (only text data).

RoBERTa-large

Roberta-large is larger than Roberta-base, which makes the model find more complex patterns. However, it is also computationally more expensive.

- **Layers**: 24
- **Hidden Units**: 1024 in each layer
- **Attention Heads**: 16
- **Parameters**: 355 million

Results and Explanation

We tried the models on ProMapEn, ProMapCz and Amazon-Walmart datasets.

ProMapEn dataset

With the Naive Bayes classifier 1.4, we gain an F1 score of 0.62. It outperforms all Bert models and the roberta base we used. This means that having a bigger context helps us understand the small domains better.

Model	F1	Acc	Prec	Rec	Name
Naive Bayes	0.62	0.71	0.55	0.71	False
MLP 1000 relu adam	0.62	0.74	0.59	0.66	False
MLP 1000 tanh	0.61	0.73	0.57	0.65	False
Logistic Regression	0.61	0.71	0.54	0.69	False
AdaBoost	0.61	0.74	0.59	0.63	False
SVC	0.6	0.7	0.53	0.69	False
Extra Trees	0.59	0.71	0.54	0.64	False
Random Forest	0.58	0.57	0.43	0.91	False
Bagging	0.57	0.75	0.65	0.51	False
XGBoost	0.56	0.59	0.43	0.8	False
KNN	0.55	0.65	0.47	0.64	False
Gradient Boosting	0.53	0.68	0.5	0.56	False
MLP 1000 relu 0.001	0.52	0.56	0.4	0.74	False
MLP 1000	0.52	0.56	0.4	0.74	False
MLP 300 200 100	0.51	0.62	0.44	0.6	False
MLP 500 500	0.49	0.46	0.36	0.8	False
Linear SVC	0.48	0.75	0.76	0.35	False
SGD	0.45	0.75	0.78	0.32	False

Table 3.20 Results for ProMapEn dataset using roberta-large (only text data).

Results on ProMapCz

The absence of improvement in Czech data means that the larger corpus has a small amount of Czech data. Naive Bayes 1.4 achieves the highest F1 score. Again, we can see that the name feature has results comparable to those of all the other features.

Model	F1	Acc	Prec	Rec	Name
Naive Bayes	0.51	0.53	0.38	0.74	False
XGBoost	0.5	0.34	0.33	1.0	True
Random Forest	0.5	0.34	0.33	1.0	True
SVC	0.5	0.37	0.34	0.95	True
Extra Trees	0.5	0.35	0.33	1.0	False
Naive Bayes	0.5	0.39	0.34	0.92	True
AdaBoost	0.5	0.34	0.33	1.0	False
MLP 500 500	0.49	0.33	0.33	1.0	True
MLP 1000	0.49	0.33	0.33	1.0	True
MLP 300 200 100	0.49	0.33	0.33	1.0	True
Logistic Regression	0.49	0.33	0.33	1.0	False
MLP 1000 tanh	0.49	0.33	0.33	1.0	True
MLP 1000 relu adam	0.49	0.33	0.33	1.0	True
MLP 1000 relu 0.001	0.49	0.33	0.33	1.0	True
AdaBoost	0.49	0.33	0.33	1.0	True
Gradient Boosting	0.49	0.33	0.33	1.0	True
Extra Trees	0.49	0.33	0.33	1.0	True
Logistic Regression	0.49	0.33	0.33	1.0	True
XGBoost	0.49	0.33	0.33	1.0	False
MLP 300 200 100	0.49	0.33	0.33	1.0	False

Table 3.21 Results for ProMapCz dataset using roberta-large (only text data).

Amazon-Walmart dataset

Using the name feature has competitive results with all other text features. The MLP classifier 1.4 has the highest F1 score of 0.8.

Model	F1	Acc	Prec	Rec	Name
MLP 300 200 100	0.8	0.85	0.77	0.82	False
AdaBoost	0.8	0.85	0.77	0.83	False
SVC	0.79	0.84	0.77	0.81	True
Random Forest	0.79	0.84	0.76	0.83	False
Naive Bayes	0.79	0.85	0.78	0.79	True
MLP 1000	0.79	0.85	0.78	0.79	True
MLP 500 500	0.79	0.85	0.78	0.79	True
MLP 300 200 100	0.79	0.85	0.79	0.79	True
MLP 1000 tanh	0.79	0.85	0.78	0.79	True
Extra Trees	0.79	0.84	0.76	0.82	False
MLP 1000 relu 0.001	0.79	0.85	0.78	0.79	True
MLP 500 500	0.79	0.84	0.76	0.82	False
AdaBoost	0.78	0.83	0.72	0.85	True
Bagging	0.78	0.84	0.81	0.74	True
KNN	0.78	0.84	0.77	0.78	False
Naive Bayes	0.78	0.83	0.74	0.83	False
Extra Trees	0.78	0.83	0.75	0.81	True
MLP 1000 relu 0.001	0.77	0.81	0.68	0.89	False
MLP 1000	0.77	0.81	0.68	0.89	False
Bagging	0.77	0.84	0.79	0.76	False

Table 3.22 Results for Amazon-Walmart dataset using roberta-large (only text data).

Amazon-Google dataset

The roberta-large is very effective on Amazon datasets. It reaches a 0.88 F1 score with Extra trees 1.4.

Again Roberta-large outperformed other bert-based models that we tried.

Model	F1	Acc	Prec	Rec	Name
Extra Trees	0.88	0.91	0.9	0.87	False
Random Forest	0.88	0.9	0.89	0.86	False
AdaBoost	0.88	0.9	0.88	0.88	False
Naive Bayes	0.87	0.89	0.89	0.84	False
KNN	0.86	0.89	0.88	0.85	False
Gradient Boosting	0.86	0.89	0.95	0.78	False
MLP 1000	0.85	0.88	0.87	0.84	False
MLP 500 500	0.85	0.88	0.87	0.84	False
MLP 300 200 100	0.85	0.89	0.92	0.8	False
MLP 1000 tanh	0.85	0.89	0.92	0.79	False
MLP 1000 relu 0.001	0.85	0.88	0.87	0.84	False
XGBoost	0.85	0.88	0.84	0.87	False
MLP 1000 tanh	0.84	0.88	0.88	0.81	True
Extra Trees	0.84	0.88	0.91	0.77	True
Bagging	0.84	0.88	0.92	0.77	True
MLP 300 200 100	0.84	0.88	0.88	0.81	True
MLP 500 500	0.84	0.88	0.88	0.81	True
MLP 1000	0.84	0.88	0.88	0.81	True
Naive Bayes	0.84	0.88	0.88	0.81	True
SVC	0.84	0.88	0.88	0.81	True

Table 3.23 Results for Amazon-Google dataset using roberta-large (only text data).

3.3.3 DistilBERT Models

DistilBERT-base-uncased

DistilBERT is based on a BERT architecture. It is a faster and smaller version of BERT.

- **Layers**: 6
- **Hidden Units**: 768 in each layer
- **Attention Heads**: 12
- **Parameters**: 66 million

Results and Explanation

We tried the models on ProMapEn, ProMapCz, Amazon-Walmart, and Amazon-Google datasets.

Results on ProMapEn dataset

We got F1 score of 0.56 with MLP classifier 1.4 with only using the name feature. This shows that roberta-large is better for this dataset than other Bert based models, however DistilBert has better results than roberta-base even though it has significantly less number of parameters.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 tanh	0.56	0.61	0.44	0.76	True
Logistic Regression	0.55	0.59	0.42	0.77	True
MLP 1000 tanh	0.55	0.61	0.44	0.71	False
Naive Bayes	0.55	0.61	0.44	0.75	True
MLP 1000 relu adam	0.54	0.61	0.44	0.7	True
MLP 1000	0.53	0.51	0.39	0.85	True
Naive Bayes	0.53	0.53	0.39	0.8	False
MLP 500 500	0.53	0.51	0.39	0.85	True
MLP 300 200 100	0.53	0.55	0.4	0.78	True
MLP 1000 relu adam	0.53	0.57	0.41	0.76	False
MLP 1000 relu 0.001	0.53	0.51	0.39	0.85	True
SVC	0.52	0.47	0.37	0.88	True
MLP 300 200 100	0.52	0.52	0.39	0.78	False
Logistic Regression	0.51	0.46	0.36	0.88	False
MLP 1000 relu 0.001	0.51	0.43	0.35	0.91	False
MLP 500 500	0.51	0.54	0.39	0.74	False
MLP 1000	0.51	0.43	0.35	0.91	False
Extra Trees	0.5	0.37	0.34	0.98	False
Random Forest	0.5	0.35	0.33	0.99	True
Extra Trees	0.5	0.4	0.34	0.91	True

Table 3.24 Results for ProMapEn dataset using distilbert-base-uncased (only text data).

ProMapCz dataset

The highest F1 that we get is 0.51 with different MLP structures. This shows that the training corpus of roberta doesn't include big czech data.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu adam	0.51	0.38	0.34	0.97	False
MLP 1000 tanh	0.51	0.4	0.35	0.93	False
MLP 500 500	0.51	0.54	0.39	0.71	True
MLP 300 200 100	0.51	0.39	0.35	0.95	True
Logistic Regression	0.51	0.54	0.39	0.71	True
XGBoost	0.5	0.47	0.36	0.81	True
Extra Trees	0.5	0.36	0.34	0.99	True
Extra Trees	0.5	0.43	0.35	0.85	False
MLP 1000 relu 0.001	0.5	0.37	0.34	0.96	False
MLP 500 500	0.5	0.36	0.33	0.97	False
MLP 1000	0.5	0.37	0.34	0.96	False
Naive Bayes	0.5	0.39	0.34	0.94	False
Logistic Regression	0.5	0.39	0.34	0.95	False
XGBoost	0.5	0.38	0.34	0.93	False
Gradient Boosting	0.5	0.39	0.34	0.94	True
MLP 1000 relu adam	0.5	0.54	0.39	0.71	True
Random Forest	0.5	0.36	0.34	0.99	True
Naive Bayes	0.5	0.41	0.35	0.92	True
MLP 1000 tanh	0.5	0.53	0.39	0.72	True
MLP 1000	0.5	0.41	0.35	0.92	True

Table 3.25 Results for ProMapCz dataset using distilbert (only text data).

Amazon-Walmart dataset

The highest F1 score we achieved is 0.78 with the AdaBoost classifier 1.4.

Model	F1	Acc	Prec	Rec	Name
AdaBoost	0.78	0.83	0.74	0.82	False
MLP 300 200 100	0.78	0.83	0.73	0.83	False
SGD	0.77	0.83	0.76	0.77	False
MLP 1000	0.77	0.82	0.71	0.84	False
MLP 1000 relu 0.001	0.77	0.82	0.71	0.84	False
Extra Trees	0.77	0.83	0.73	0.82	False
Naive Bayes	0.77	0.82	0.71	0.82	False
Linear SVC	0.77	0.83	0.76	0.77	False
MLP 500 500	0.76	0.81	0.71	0.81	True
MLP 300 200 100	0.76	0.82	0.73	0.8	True
MLP 1000 tanh	0.76	0.82	0.73	0.8	True
MLP 1000	0.76	0.81	0.71	0.81	True
MLP 1000 relu 0.001	0.76	0.81	0.71	0.81	True
Naive Bayes	0.76	0.82	0.73	0.8	True
SVC	0.76	0.82	0.72	0.81	True
MLP 500 500	0.76	0.81	0.7	0.84	False
Linear SVC	0.76	0.83	0.76	0.76	True
Logistic Regression	0.76	0.82	0.72	0.81	True
Logistic Regression	0.76	0.8	0.69	0.83	False
Bagging	0.76	0.83	0.77	0.76	False

Table 3.26 Results for Amazon-Walmart dataset using distilbert (only text data).

Amazon-Google dataset

Distilbert achieved similar results to roberta-large on the Amazon-Google dataset using MLP classifier 1.4

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu adam	0.87	0.89	0.88	0.85	False
XGBoost	0.86	0.88	0.86	0.85	False
Random Forest	0.86	0.89	0.87	0.84	False
AdaBoost	0.86	0.89	0.9	0.82	False
MLP 1000 relu 0.001	0.86	0.9	0.92	0.82	False
MLP 300 200 100	0.86	0.89	0.91	0.82	False
MLP 500 500	0.86	0.9	0.93	0.81	False
MLP 1000	0.86	0.9	0.92	0.82	False
Extra Trees	0.86	0.89	0.93	0.8	False
KNN	0.85	0.88	0.88	0.81	False
SVC	0.84	0.86	0.82	0.85	False
Gradient Boosting	0.84	0.88	0.9	0.79	False
Bagging	0.84	0.88	0.89	0.8	False
Naive Bayes	0.84	0.87	0.83	0.84	False
MLP 1000 tanh	0.83	0.86	0.83	0.83	False
Logistic Regression	0.83	0.86	0.82	0.83	False
Linear SVC	0.82	0.86	0.84	0.8	False
SVC	0.81	0.85	0.86	0.77	True
Naive Bayes	0.81	0.85	0.82	0.8	True
MLP 1000	0.81	0.85	0.82	0.8	True

Table 3.27 Results for Amazon-Google dataset using distilbert (only text data).

3.4 Sentence Transformers

Sentence Transformers [27] is a family of models specifically designed to create high-quality sentence embeddings. These models are fine-tuned on Bert and roberta to optimize the embeddings for sentence-level tasks.

3.4.1 all-MiniLM-L6-v2

The all-MiniLM-L6-v2 model balances performance and computational efficiency, making it suitable for applications requiring quick and accurate sentence embeddings.

- **Layers**: 6
- **Hidden Units**: 384
- **Attention Heads**: 16
- **Parameters**: 22 million

Results and Explanation

We tried the models on ProMapEn, ProMapCz, Amazon-Walmart and Amazon-Google datasets.

Results on ProMapEn

We get promising results by using sentence transformers. The highest F1 value is 0.61 with the Naive Bayes 1.4 model using all text features. Interestingly, having 16 times fewer parameters, it performs as well as roberta-large.

Model	F1	Acc	Prec	Rec	Name
Naive Bayes	0.61	0.68	0.51	0.75	False
Logistic Regression	0.61	0.69	0.51	0.76	False
SVC	0.6	0.68	0.51	0.72	False
SVC	0.6	0.66	0.48	0.78	True
MLP 1000 tanh	0.6	0.68	0.5	0.76	False
MLP 1000 relu adam	0.6	0.66	0.48	0.78	True
MLP 500 500	0.59	0.63	0.46	0.81	True
MLP 300 200 100	0.59	0.61	0.45	0.85	True
MLP 1000	0.59	0.63	0.46	0.82	True
Naive Bayes	0.59	0.64	0.47	0.81	True
MLP 1000 tanh	0.59	0.65	0.48	0.78	True
Logistic Regression	0.59	0.65	0.47	0.79	True
MLP 1000 relu 0.001	0.59	0.63	0.46	0.82	True
AdaBoost	0.58	0.61	0.45	0.83	True
AdaBoost	0.56	0.55	0.41	0.86	False
Extra Trees	0.55	0.58	0.42	0.78	False
MLP 1000 relu adam	0.55	0.59	0.43	0.78	False
Random Forest	0.55	0.53	0.4	0.87	False
Random Forest	0.54	0.49	0.38	0.91	True
Bagging	0.53	0.69	0.53	0.52	False

Table 3.28 Results for ProMapEn dataset using all-MiniLM-L6-v2 (only text data).

ProMapCz dataset

The highest F1 that we get is 0.54 with different MLP structures. This is the best bet-based model so far, outperforming even Roberta-large.

Model	F1	Acc	Prec	Rec	Name
MLP 300 200 100	0.54	0.57	0.42	0.78	False
MLP 1000	0.54	0.63	0.45	0.67	True
MLP 300 200 100	0.54	0.52	0.39	0.86	True
Extra Trees	0.54	0.5	0.38	0.89	True
MLP 1000 relu 0.001	0.54	0.63	0.45	0.67	True
MLP 1000 tanh	0.53	0.45	0.37	0.96	False
MLP 1000	0.53	0.59	0.42	0.72	False
Naive Bayes	0.53	0.44	0.37	0.94	False
MLP 1000 relu adam	0.53	0.47	0.37	0.91	False
MLP 1000 relu 0.001	0.53	0.59	0.42	0.72	False
Logistic Regression	0.53	0.44	0.37	0.95	False
Logistic Regression	0.53	0.63	0.45	0.65	True
XGBoost	0.53	0.51	0.39	0.85	True
MLP 1000 relu adam	0.53	0.63	0.46	0.63	True
MLP 1000 tanh	0.53	0.62	0.45	0.65	True
MLP 500 500	0.53	0.61	0.44	0.68	True
Naive Bayes	0.53	0.63	0.45	0.65	True
SVC	0.53	0.63	0.45	0.65	True
Random Forest	0.53	0.47	0.37	0.9	True
Gradient Boosting	0.53	0.44	0.37	0.94	True

Table 3.29 Results for ProMapCz dataset using all-MiniLM-L6-v2 (only text data).

Amazon-Walmart dataset

With Extra Trees 1.4, we get an F1 value of 0.9.

Model	F1	Acc	Prec	Rec	Name
Extra Trees	0.9	0.92	0.86	0.94	False
Bagging	0.9	0.92	0.87	0.93	False
MLP 1000 relu adam	0.89	0.92	0.89	0.89	False
Linear SVC	0.89	0.92	0.87	0.91	False
MLP 300 200 100	0.89	0.91	0.86	0.91	False
SVC	0.89	0.92	0.86	0.93	False
Random Forest	0.89	0.92	0.86	0.92	False
Logistic Regression	0.89	0.91	0.85	0.92	False
SGD	0.88	0.92	0.92	0.84	False
AdaBoost	0.88	0.91	0.83	0.93	False
MLP 1000 tanh	0.88	0.91	0.85	0.91	False
Naive Bayes	0.88	0.91	0.83	0.94	False
KNN	0.88	0.91	0.86	0.89	False
SGD	0.88	0.91	0.86	0.89	True
Linear SVC	0.88	0.91	0.86	0.89	True
Gradient Boosting	0.88	0.91	0.86	0.89	False
AdaBoost	0.88	0.91	0.86	0.89	True
Extra Trees	0.88	0.91	0.86	0.9	True
MLP 1000	0.87	0.9	0.83	0.92	False
MLP 1000 relu 0.001	0.87	0.9	0.83	0.92	False

Table 3.30 Results for Amazon-Walmart dataset using all-MiniLM-L6-v2 (only text data).

Amazon-Google dataset

This case is also the best one compared to previously tried models. The training is done with all text features. All-MiniLM-L6-v2 gives us the greatest performance on the Amazon-Google dataset, using all textual data and an MLP classifier 1.4 or Random Forest 1.4.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu adam	0.99	0.99	0.98	0.99	False
Random Forest	0.99	0.99	0.99	0.98	False
Naive Bayes	0.99	0.99	0.98	0.99	False
AdaBoost	0.99	0.99	0.99	0.98	False
MLP 1000 tanh	0.98	0.99	0.98	0.99	False
SVC	0.98	0.99	0.98	0.99	False
KNN	0.98	0.99	0.98	0.99	False
MLP 1000	0.98	0.99	0.98	0.98	False
MLP 300 200 100	0.98	0.98	0.98	0.98	False
XGBoost	0.98	0.98	0.97	0.98	True
Logistic Regression	0.98	0.98	0.97	0.98	True
MLP 1000 relu 0.001	0.98	0.99	0.98	0.98	False
Gradient Boosting	0.98	0.98	0.97	0.98	False
Extra Trees	0.98	0.99	0.98	0.99	False
Bagging	0.98	0.99	0.98	0.99	False
Linear SVC	0.98	0.99	0.98	0.98	False
Logistic Regression	0.98	0.99	0.98	0.99	False
XGBoost	0.98	0.98	0.98	0.98	False
SGD	0.98	0.98	0.99	0.96	True
MLP 1000 relu adam	0.98	0.98	0.97	0.98	True

Table 3.31 Results for Amazon-Google dataset using all-MiniLM-L6-v2 (only text data).

3.4.2 all-mpnet-base-v2

The architecture of all-mpnet-base-v2 is as follows

- **Layers**: 12
- **Hidden Units**: 768
- **Attention Heads**: 12
- **Parameters**: 110 million

Results and Explanation

ProMapEn dataset

With Naive Bayes, we got an F1 score of 0.58, and all-MiniLM-L6-v2 outperforms this model with a 0.61 F1 score.

Model	F1	Acc	Prec	Rec	Name
Naive Bayes	0.58	0.69	0.52	0.64	False
MLP 1000 relu adam	0.57	0.67	0.49	0.67	True
MLP 300 200 100	0.57	0.68	0.51	0.63	True
MLP 1000 tanh	0.57	0.63	0.46	0.76	False
MLP 1000 relu adam	0.57	0.59	0.44	0.84	False
Logistic Regression	0.57	0.67	0.49	0.67	True
Logistic Regression	0.57	0.62	0.45	0.76	False
AdaBoost	0.56	0.67	0.5	0.63	True
MLP 1000 tanh	0.56	0.65	0.47	0.69	True
MLP 500 500	0.56	0.65	0.47	0.7	True
MLP 1000	0.56	0.64	0.46	0.71	True
Naive Bayes	0.56	0.65	0.48	0.69	True
MLP 1000 relu 0.001	0.56	0.64	0.46	0.71	True
SVC	0.55	0.66	0.48	0.65	True
AdaBoost	0.55	0.56	0.41	0.82	False
Extra Trees	0.54	0.56	0.41	0.78	False
SVC	0.54	0.55	0.4	0.82	False
Random Forest	0.54	0.59	0.42	0.74	False
Extra Trees	0.53	0.45	0.37	0.95	True
MLP 500 500	0.53	0.54	0.4	0.8	False

Table 3.32 Results for ProMapEn dataset using all-MiniLM-L6-v2 (only text data).

ProMapCz dataset

The highest F1 we get with Random Forest is 0.53. For this dataset, too, the all-MiniLM-L6-v2 has better results.

Model	F1	Acc	Prec	Rec	Name
Random Forest	0.53	0.58	0.42	0.72	False
MLP 500 500	0.53	0.52	0.39	0.82	False
MLP 1000 relu adam	0.53	0.63	0.45	0.63	False
Extra Trees	0.53	0.56	0.4	0.76	False
Gradient Boosting	0.52	0.5	0.38	0.82	False
XGBoost	0.51	0.43	0.35	0.91	True
MLP 500 500	0.51	0.42	0.36	0.93	True
Random Forest	0.51	0.51	0.38	0.79	True
Logistic Regression	0.51	0.42	0.36	0.93	True
Logistic Regression	0.51	0.51	0.38	0.77	False
MLP 300 200 100	0.51	0.44	0.36	0.89	True
MLP 1000 tanh	0.51	0.42	0.35	0.93	True
MLP 1000 relu adam	0.51	0.43	0.36	0.92	True
MLP 1000 relu 0.001	0.51	0.45	0.36	0.86	False
MLP 1000 relu 0.001	0.51	0.38	0.34	0.98	True
Naive Bayes	0.51	0.42	0.36	0.93	True
MLP 1000 tanh	0.51	0.45	0.36	0.88	False
AdaBoost	0.51	0.4	0.35	0.94	True
MLP 1000	0.51	0.45	0.36	0.86	False
Naive Bayes	0.51	0.44	0.36	0.91	False

Table 3.33 Results for ProMapCz dataset using all-mpnet-base-v2 (only text data).

Amazon-Google dataset

All-mpnet-base-v2 proves to be very effective compared to most of the models we have tried so far. The interesting part is that using only the name feature has the highest F1 score with the Stochastic Gradient Descent Classifier 1.4. In other models, using all of the text features had better results than using only the name feature.

Model	F1	Acc	Prec	Rec	Name
SGD	0.98	0.98	0.97	0.98	True
MLP 1000	0.98	0.98	0.97	0.98	True
MLP 500 500	0.98	0.98	0.97	0.98	True
SVC	0.98	0.98	0.97	0.98	True
MLP 300 200 100	0.98	0.98	0.97	0.98	True
Logistic Regression	0.98	0.98	0.97	0.98	True
MLP 1000 tanh	0.98	0.98	0.97	0.98	True
MLP 1000 relu adam	0.98	0.98	0.97	0.98	True
MLP 1000 relu 0.001	0.98	0.98	0.97	0.98	True
Bagging	0.98	0.99	0.98	0.98	False
Extra Trees	0.98	0.98	0.97	0.99	False
AdaBoost	0.98	0.99	0.98	0.99	False
MLP 1000 relu 0.001	0.98	0.98	0.99	0.97	False
MLP 1000 relu adam	0.98	0.98	0.97	0.99	False
MLP 1000 tanh	0.98	0.98	0.96	0.99	False
MLP 1000	0.98	0.98	0.99	0.97	False
Naive Bayes	0.98	0.98	0.97	0.99	False
KNN	0.98	0.98	0.97	0.98	False
SVC	0.98	0.99	0.98	0.99	False
Random Forest	0.98	0.98	0.98	0.98	False

Table 3.34 Results for Amazon-Google dataset using all-mpnet-base-v2 (only text data).

3.4.3 stsb-roberta-large

The architecture of stsb-roberta-large is as follows

- **Layers**: 24
- **Hidden Units**: 1024
- **Attention Heads**: 16
- **Parameters**: 355 million

Results and Explanation

We tried the models on ProMapEn and ProMapCz datasets. The reason is that stsb-roberta-large has promising results on ProMap datasets.

ProMapEn dataset

stsb-roberta-large gives us the highest F1 score for the ProMapEn dataset out of all the models we have tried. The F1 score is 0.65 with only using the name feature with MLP classifier 1.4.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu adam	0.65	0.78	0.67	0.62	True
Naive Bayes	0.65	0.78	0.68	0.62	True
MLP 1000 tanh	0.65	0.78	0.67	0.62	True
SVC	0.64	0.77	0.66	0.62	True
Logistic Regression	0.64	0.77	0.66	0.62	True
MLP 1000	0.63	0.77	0.64	0.62	True
MLP 500 500	0.63	0.75	0.61	0.65	True
MLP 1000 relu 0.001	0.63	0.77	0.64	0.62	True
Naive Bayes	0.62	0.71	0.55	0.71	False
AdaBoost	0.61	0.74	0.59	0.63	False
Logistic Regression	0.61	0.71	0.54	0.69	False
SVC	0.6	0.7	0.53	0.69	False
MLP 1000 tanh	0.6	0.67	0.5	0.77	False
SGD	0.59	0.7	0.53	0.67	False
MLP 300 200 100	0.59	0.68	0.5	0.71	True
Extra Trees	0.59	0.7	0.53	0.67	False
Random Forest	0.59	0.75	0.63	0.54	False
AdaBoost	0.58	0.66	0.48	0.73	True
MLP 1000 relu adam	0.57	0.59	0.43	0.82	False
Bagging	0.57	0.75	0.65	0.51	False

Table 3.35 Results for ProMapEn dataset using stsb-roberta-large (only text data).

ProMapCz dataset

The same as for the ProMapEn dataset, the model performed the highest F1 score compared to previous models. We get a 0.64 F1 score using all text features with the MLP classifier.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu adam	0.64	0.68	0.51	0.86	False
Naive Bayes	0.64	0.7	0.52	0.83	False
Logistic Regression	0.63	0.7	0.53	0.78	False
SVC	0.63	0.66	0.49	0.9	False
Extra Trees	0.63	0.67	0.49	0.87	False
AdaBoost	0.63	0.69	0.52	0.79	False
MLP 1000 tanh	0.63	0.7	0.52	0.8	False
MLP 1000 relu adam	0.62	0.68	0.5	0.8	True
Logistic Regression	0.62	0.68	0.5	0.8	True
Naive Bayes	0.62	0.68	0.5	0.8	True
MLP 500 500	0.62	0.68	0.51	0.78	True
Random Forest	0.62	0.68	0.51	0.82	False
MLP 1000 relu 0.001	0.61	0.68	0.51	0.78	True
MLP 1000	0.61	0.68	0.51	0.78	True
MLP 1000 tanh	0.61	0.67	0.5	0.81	True
SVC	0.61	0.68	0.5	0.78	True
AdaBoost	0.61	0.66	0.48	0.82	True
MLP 300 200 100	0.61	0.67	0.5	0.8	True
Extra Trees	0.6	0.61	0.45	0.88	True
Bagging	0.6	0.73	0.58	0.61	False

Table 3.36 Results for ProMapCz dataset using all-mpnet-base-v2 (only text data).

3.5 Natural Language Inference (NLI) Models

Models for Natural Language Inference (NLI) [27] are vital when it comes to understanding and foretelling the connections between two sentences which is essential for various NLP tasks. These models have been trained on huge datasets like the Multi-Genre Natural Language Inference (MNLI) dataset in order to comprehend if one sentence entails, contradicts, or is neutral towards another. Here we will discuss the following NLI models: `nli-roberta-base-v2`, `nli-bert-base`, `nli-distilroberta-base-v2`, and `nli-mpnet-base-v2`.

3.5.1 nli-roberta-base-v2

The model `nli-roberta-base-v2` is a fine-tuned version of RoBERTa architecture optimised for natural language inference tasks. This particular model is trained on the same BERT model with a bigger training dataset, extended training time, and dynamic masking strategies employed. It has been fine-tuned to the MNLI data set, which uses different genres to establish relationships between sentence pairs (entailment, contradiction, or neutral).

3.5.2 nli-bert-base

The `nli-bert-base` model is based on the BERT architecture. It is fine-tuned for natural language inference tasks using datasets like MNLI to identify entailment, contradiction, and neutrality

Use Cases The `nli-bert-base` model is suited for:

- **Textual Entailment**: Evaluating logical relationships between sentence pairs.
- **Sentiment Analysis**: Guessing sentiment by comprehending context.
- **Dialog Systems**: Producing suitable responses given a context.

Performance Fine-tuned on NLI datasets, the nli-bert-base model excels in understanding and predicting sentence relationships.

3.5.3 nli-distilroberta-base-v2

NLI-DistilRoBERTA-Base-V2 is a distilled version of RoBERTa, which is optimized for efficiency while still performing well on NLI tasks. Size reduction through distillation makes it faster and more suitable for deployment into resource-constrained environments. Using databases such as MNLI, it has been perfected to perform best in sentence pair classification tasks.

Use Cases The `nli-distilroberta-base-v2` model is effective for:

- **Real-Time Inference**: Quick check of sentence pairs
- **Mobile Applications**: Working on limited devices efficiently
- **Text Classification**: Classifying relationships between sentences.

Data Used This model was trained on the MNLI dataset.

3.5.4 nli-mpnet-base-v2

The `nli-mpnet-base-v2` model leverages the MPNet architecture, which combines masked language modeling and permuted language modeling to capture dependencies and relationships between words and sentences. It has been fine-tuned on the MNLI dataset to enhance its performance in natural language inference tasks.

Use Cases The `nli-mpnet-base-v2` model can be used for:

- **Textual Entailment**: Assessing entailment, contradiction, and neutrality between sentences.
- **Semantic Similarity**: Measuring the similarity between sentence pairs.
- **Information Retrieval**: Enhancing the retrieval of relevant documents based on sentence queries.

We used nli-roberta-base-v2, nli-bert-base, nli-distilroberta-base-v2, nli-mpnet-base-v2 processing methods.

Results We tried ProMapEn and ProMapCz datasets with the following preprocessing methods, which have interesting and promising results.

ProMapEn dataset

The highest F1 score gives nli-mpnet-base-v2. The following MLP architecture 1.4 is used to get the result.

Model	F1	Acc	Prec	Rec	Name	Processed Model
MLP 1000 relu 0.001	0.61	0.69	0.52	0.73	True	mpnet-base-v2
Logistic Regression	0.61	0.68	0.51	0.75	True	mpnet-base-v2
MLP 1000	0.61	0.69	0.52	0.73	True	mpnet-base-v2
MLP 1000 relu adam	0.6	0.65	0.47	0.83	False	mpnet-base-v2
MLP 1000 tanh	0.6	0.65	0.47	0.81	False	mpnet-base-v2
Naive Bayes	0.6	0.68	0.5	0.75	True	mpnet-base-v2
MLP 300 200 100	0.6	0.68	0.5	0.75	True	mpnet-base-v2
MLP 1000 tanh	0.6	0.68	0.5	0.75	True	mpnet-base-v2
MLP 1000 relu adam	0.6	0.68	0.5	0.75	True	mpnet-base-v2
Naive Bayes	0.59	0.66	0.49	0.75	False	mpnet-base-v2
Logistic Regression	0.59	0.65	0.48	0.78	False	mpnet-base-v2
MLP 500 500	0.59	0.67	0.49	0.75	True	mpnet-base-v2
MLP 1000 relu adam	0.58	0.73	0.58	0.57	True	bert-base
AdaBoost	0.58	0.68	0.5	0.7	True	mpnet-base-v2
MLP 1000 tanh	0.57	0.72	0.56	0.57	True	bert-base
MLP 300 200 100	0.57	0.72	0.58	0.56	True	bert-base
AdaBoost	0.57	0.56	0.42	0.9	False	mpnet-base-v2
MLP 500 500	0.57	0.72	0.56	0.57	True	bert-base
Extra Trees	0.57	0.54	0.41	0.92	True	mpnet-base-v2
Logistic Regression	0.57	0.71	0.56	0.57	True	bert-base

Table 3.37 Top 20 results for ProMapEn data using NLI models (only text data).

ProMapCz dataset

nli-distilroberta-base-v2 has a better context of the Czech language than other NLI models. The F1 score is 0.58 using Extra Trees.

Model	F1	Acc	Prec	Rec	Name	Processed Model
Extra Trees	0.58	0.66	0.48	0.73	False	distilroberta-base-v2
MLP 300 200 100	0.58	0.66	0.49	0.71	False	distilroberta-base-v2
MLP 300 200 100	0.57	0.76	0.67	0.5	True	bert-base
MLP 300 200 100	0.57	0.72	0.57	0.57	False	bert-base
Logistic Regression	0.57	0.75	0.66	0.5	True	bert-base
MLP 1000	0.57	0.59	0.43	0.83	False	mpnet-base-v2
SVC	0.57	0.75	0.66	0.5	True	bert-base
Naive Bayes	0.57	0.68	0.51	0.63	False	distilroberta-base-v2
MLP 300 200 100	0.57	0.69	0.52	0.62	False	mpnet-base-v2
MLP 500 500	0.57	0.75	0.66	0.5	True	bert-base
MLP 1000 relu 0.001	0.57	0.59	0.43	0.83	False	mpnet-base-v2
MLP 1000	0.57	0.75	0.66	0.5	True	bert-base
MLP 1000 tanh	0.57	0.75	0.66	0.5	True	bert-base
AdaBoost	0.57	0.76	0.67	0.5	True	bert-base
Random Forest	0.57	0.67	0.49	0.67	False	distilroberta-base-v2
Extra Trees	0.57	0.7	0.53	0.61	False	mpnet-base-v2
Bagging	0.57	0.76	0.67	0.5	True	bert-base
MLP 1000 relu adam	0.57	0.71	0.55	0.58	False	mpnet-base-v2
MLP 1000 relu 0.001	0.57	0.75	0.66	0.5	True	bert-base
MLP 1000 relu adam	0.57	0.75	0.66	0.5	True	bert-base

Table 3.38 Top 20 results for ProMapCz data using NLI models

3.6 Specialized Models

3.6.1 abbasgolestani/ag-nli-DeTS-sentence-similarity-v1

The `abbasgolestani/ag-nli-DeTS-sentence-similarity-v1` model [28] is a Cross-Encoder for sentence similarity tasks. It predicts a similarity score between two sentences, ranging from 0 (not similar) to 1 (very similar). This model was trained using the SentenceTransformers Cross-Encoder class on six different NLI datasets, including Multi-NLI and NLI Fever datasets.

Use Cases This model can be used for various NLP applications, such as:

- **Semantic Search**: Finding sentences or documents with similar meanings.
- **Paraphrase Detection**: Identifying sentences that convey the same meaning.
- **Textual Similarity**: Measuring the similarity between different texts for applications like duplicate detection.

Performance and results The `abbasgolestani/ag-nli-DeTS-sentence-similarity-v1` model has significantly improved sentence similarity benchmarks, making it a good choice for tasks requiring precise and context-aware embeddings.

Results We experimented on ProMapEn and ProMapCz datasets. The reason is that ag-nli-DeTS-sentence-similarity-v1 gave superior results on both datasets, and we got the highest F1 scores out of all preprocessing models that we have tried so far.

ProMapEn dataset

The model gives the highest F1 value for the ProMapEn dataset. We get 0.72 with Extra Trees 1.4.

Model	F1	Acc	Prec	Rec	Name
SVC	0.72	0.81	0.7	0.73	False
Extra Trees	0.72	0.8	0.66	0.79	False
Extra Trees	0.71	0.8	0.67	0.76	False
MLP 1000	0.71	0.78	0.62	0.83	False
AdaBoost	0.71	0.8	0.67	0.75	False
MLP 1000 relu 0.001	0.71	0.78	0.62	0.83	False
SVC	0.71	0.8	0.68	0.74	False
Random Forest	0.7	0.78	0.63	0.78	False
SVC	0.7	0.78	0.64	0.76	False
MLP 300 200 100	0.69	0.77	0.63	0.76	False
AdaBoost	0.69	0.77	0.62	0.78	False
MLP 500 500	0.69	0.76	0.6	0.81	False
MLP 300 200 100	0.69	0.77	0.61	0.8	False
Random Forest	0.69	0.77	0.61	0.8	False
MLP 1000 tanh	0.69	0.77	0.62	0.77	False
MLP 1000 relu adam	0.69	0.77	0.62	0.77	False
Logistic Regression	0.69	0.78	0.64	0.76	False
XGBoost	0.69	0.79	0.68	0.69	False
MLP 1000	0.68	0.76	0.6	0.77	False
MLP 500 500	0.68	0.76	0.6	0.77	False

Table 3.39 Results for ProMapEn data (only text data).

ProMapCz dataset

The model also performs well on ProMapCz data. With the support vector machines classifier, we get a 0.68 F1 score.

Model	F1	Acc	Prec	Rec	Name
MLP 1000 relu 0.001	0.68	0.76	0.6	0.8	False
MLP 1000	0.68	0.76	0.6	0.8	False
AdaBoost	0.67	0.74	0.57	0.81	False
Bagging	0.67	0.75	0.58	0.79	False
MLP 300 200 100	0.67	0.73	0.56	0.82	False
MLP 500 500	0.67	0.73	0.56	0.84	False
SVC	0.67	0.76	0.6	0.76	False
Extra Trees	0.66	0.73	0.56	0.79	True
MLP 1000 relu adam	0.66	0.74	0.57	0.79	False
MLP 1000 tanh	0.66	0.74	0.58	0.77	False
Logistic Regression	0.66	0.75	0.6	0.72	False
Logistic Regression	0.66	0.74	0.58	0.78	True
Bagging	0.66	0.76	0.62	0.71	True
SVC	0.66	0.74	0.57	0.79	True
MLP 1000 relu 0.001	0.66	0.73	0.57	0.8	True
MLP 1000 relu adam	0.66	0.73	0.57	0.8	True
MLP 1000 tanh	0.66	0.73	0.57	0.8	True
AdaBoost	0.66	0.74	0.58	0.78	True
MLP 300 200 100	0.66	0.74	0.57	0.79	True
MLP 500 500	0.66	0.73	0.57	0.8	True

Table 3.40 Results for ProMapCz data using (only text data).

3.7 Fine-Tuning BERT for Sentence Similarity

In this section, we discuss our approach to fine-tuning a BERT-based model for Product Mapping. We have tried two different approaches for fine-tuning.

3.7.1 First Fine-Tuned model

The architecture of the fine-tuned BERT model for sentence similarity consists of several key components:

- **Input Layers**:
  - **Input IDs**: Encoded token IDs generated by the BERT tokenizer, with a shape of (max_length,) and data type `tf.int32`.
  - **Attention Masks**: Binary masks indicating which tokens should be attended to, with a shape of (max_length,) and data type `tf.int32`.
  - **Token Type IDs**: Binary masks identifying different sequences within the input data, with a shape of (max_length,) and data type `tf.int32`.
- **BERT Model**:
  - A pretrained BERT model (`bert-base-uncased`) is used. The BERT model's layers are frozen to retain its pretrained features.
- **Bidirectional LSTM**:

  - A Bidirectional LSTM layer with 64 units is applied to the sequence output from the BERT model. This layer has `return_sequences=True` to output the full sequence.

- **Hybrid Pooling**:
  - **Global Average Pooling**: The average value of the features across the sequence.
  - **Global Max Pooling**: The maximum value of the features across the sequence.
  - The outputs of the pooling layers are concatenated.

- **Dropout Layer**:
  - A dropout layer with a rate of 0.2 is added to prevent overfitting.

- **Dense Layer**:
  - Sigmoid activation function. It is used to produce the final binary similarity score.

Model Compilation

The model is compiled with the following configuration:

- **Optimizer**: Adam optimizer.
- **Loss Function**: Binary cross-entropy loss.
- **Metrics**: Accuracy, F1 score, precision, and recall.

Results The fine-tuned model achieved good results on ProMapCz, Amazon-Walmart, and Amazon-Google datasets. However, it was problematic with ProMapEn because it failed to avoid overfitting.

	Accuracy	F1	Precision	Recall
ProMapEN	0.73	0.46	0.73	0.39
ProMapCZ	0.78	0.72	0.69	0.79
AmazonWalmart	0.96	0.93	0.94	0.94
AmazonGoogle	0.99	0.98	0.99	0.98

Table 3.41 Results of Fine-tuned Bert (only text data).

The biggest improvement was in Czech data when we compared the results of using Bert models without fine-tuning.

3.7.2 Fine-Tuning BERT for Product Matching with Triplet-Loss

This section discusses our approach to fine-tuning a BERT-based model for Product Mapping, leveraging triplet loss for similarity learning. The architecture and the training methodology are inspired by the work presented in "BERT-based similarity learning for Product Mapping" by Tracz et al. (2020) [26]. We extend their approach by incorporating advanced encoder architectures and custom batch construction strategies to enhance model performance.

Model Architecture and Triplet Loss

The `ProductMatchingModel` utilizes the `bert-large-cased` model from the Hugging Face Transformers library as its backbone. The architecture includes:

- **BERT Model**: A pretrained `bert-large-cased` model provides contextual embeddings for input text sequences.
- **Embedding Layer**: A linear layer reduces the dimensionality of the BERT pooled output to 768 dimensions.
- **Trainable Parameters**: All parameters of the BERT model are set to be trainable, allowing the model to adapt specifically to the Product Mapping task.

To address the Product Mapping problem, we use a similarity learning approach with triplet loss. The triplet loss function encourages the model to learn an embedding space where:

- Similar products (anchor and positive) are closer to each other.
- Dissimilar products (anchor and negative) are further apart.

The triplet loss is defined as:

$$L(o, p^+, p^-) = \max(0, m + d(E_\theta(o), E_\phi(p^+)) - d(E_\theta(o), E_\phi(p^-))),$$

where o is the anchor, p^+ is the positive (matching product), p^- is the negative (non-matching product), d is the cosine distance, and m is a margin hyperparameter. This loss helps create a robust embedding space for Product Mapping, addressing issues like data heterogeneity and varying levels of data quality [26].

Finding Negative Samples for Triplet Loss

We select negatives from the same category as the anchor and positive samples. This ensures that the model learns to distinguish between visually similar products that are not actual matches, thus enhancing its ability to identify matching products correctly.

Problem Addressed by Triplet Loss

Triplet loss is particularly effective for Product Mapping as it optimizes the distance between matched and non-matched products in the embedding space. This method is crucial for handling the high heterogeneity and large volume of products in e-commerce datasets. By minimizing the triplet loss, the model learns to distinguish between similar and dissimilar products more accurately, improving the overall performance of the Product Mapping system.

Models Utilized and Training with Triplet Loss

We utilized several BERT-based models and fine-tuned them using the triplet loss approach. The models include:

- **BERT-base-uncased**
- **BERT-large-uncased**
- **DistilBERT-base-uncased**
- **nli-roberta-base-v2**
- **nli-bert-base**
- **nli-distilroberta-base-v2**
- **nli-mpnet-base-v2**

The triplet loss function encourages the model to learn an embedding space where similar products (anchor and positive) are closer to each other while dissimilar products (anchor and negative) are further apart. This method is particularly effective for Product Mapping, as it optimizes the distance between matched and non-matched products in the embedding space.

3.7.3 Performance Evaluation

The models were evaluated using the F1 score, measuring a test's accuracy. The F1 scores for the models trained with triplet loss on various datasets are as follows:

- **ProMapEn**: 0.54
- **ProMapCz**: 0.53
- **Amazon-Google**: 0.77
- **Amazon-Walmart**: 0.61

These results highlight the effectiveness of the triplet loss approach in improving the model's performance on different product-matching datasets.

The training process involved:

- **Batch Construction**: Using strategies such as category hard (CH) and batch hard (BH) to select triplets that contribute effectively to the learning process.
- **Pretraining and Fine-Tuning**: Pretraining the models on a large corpus followed by fine-tuning with triplet loss on the Product Mapping dataset.
- **Evaluation**: Measuring the performance using metrics such as accuracy and F1 score to ensure robust Product Mapping capabilities.

By employing these strategies and models, we significantly improved the performance of the Product Mapping system, making it more efficient and accurate in identifying similar products across diverse e-commerce platforms [26].

3.8 Conclusion

The evaluation of traditional models, transformer-based models, and fine-tuned BERT architectures has highlighted the strengths and weaknesses of each approach in the context of Product Mapping. Traditional models like TF-IDF and Word2Vec demonstrated consistent performance across various datasets, with TF-IDF showing adaptability to new domains. Although more complex, advanced models like BERT and RoBERTa provided nuanced embeddings that effectively captured contextual information. The following are the best F1 results we get using text data.

- **ProMapEn** : 0.72 3.6.1
  - **Processing method**: abbasgolestani/ag-nli-DeTS-sentence-similarity-v1
  - **Classifier**: SVC
- **ProMapCz**: 0.72 3.7.1
  - **Model**: Fine-Tuned bert
- **Amazon-Google**: 0.99 3.7.1
  - **Model**: Fine-Tuned bert
- **Amazon-Walmart**: 0.93 3.7.1
  - **Model**: Fine-Tuned bert

Fine-tuning BERT models showed big improvements in Product Mapping. This approach addressed data heterogeneity and varying quality challenges, enhancing the model's ability to distinguish between similar and dissimilar products. However, despite these advancements, the overall performance varied across datasets, emphasizing the need for domain-specific adjustments. The comprehensive evaluation underscores the importance of choosing the suitable model and fine-tuning strategy to achieve optimal results in Product Mapping applications.

4 Combining Text and Image Data

This part describes combining text and image data for product mapping tasks on ProMapEn and ProMapCz datasets. Integrating multimodal data is meant to capitalize on the power of textual and visual features and improve the performance levels of product-mapping models. Amazon-Google and Amazon-Walmart are not involved as they do not contain any image data.

4.0.1 Methodology

We utilized the subsequent methodologies for combining text and image data:

- **Text Feature Extraction**: We have already seen that ag-nli-DeTS-sentence-similarity-v1 is the best, not finetuned, model for extracting texts. Therefore, we will use this model for text extraction.
- **Image Feature Extraction**: Comparison between pre-trained image models combining the text data.
- **Feature Concatenation**: To know which works better among all these image models, we compute cosine similarities between text features extracted by ag-nli-DeTS-sentence-similarity-v1 and images.
- **Classification Models**: We trained the data with models like Logistic Regression, Support Vector Classifier (SVC), Multi-Layer Perceptron (MLP), etc.

4.0.2 Results on ProMapEn Dataset

The results of combining text and image data on the ProMapEn dataset are presented in Table 4.1. The highest F1 score for ProMapEn is 0.73, using all the features. Let us note that it is the highest score for the dataset that we got. The pre-trained image model is efficientnet, and the classifier is MLP 1.4.

We can also see that top models used all of the text features for training. This means that using only the name feature with image data does not have

The interesting point is that initially, VGG16 performed the worst out of all image-pre-trained models. However, when combined with the text feature, it gives competitive results.

Model	F1	Acc	Prec	Rec	Name	Image
MLP 1000 tanh	0.73	0.8	0.66	0.8	False	efficientnet
SVC	0.72	0.81	0.7	0.73	False	vgg
Extra Trees	0.72	0.8	0.66	0.79	False	vgg
Extra Trees	0.72	0.81	0.68	0.77	False	efficientnet
AdaBoost	0.71	0.8	0.67	0.75	False	resnet
SVC	0.71	0.8	0.68	0.74	False	resnet
AdaBoost	0.71	0.8	0.69	0.72	False	efficientnet
Extra Trees	0.71	0.8	0.67	0.76	False	resnet
Random Forest	0.71	0.8	0.66	0.78	False	efficientnet
Random Forest	0.7	0.79	0.67	0.73	False	MaxOutputEnsemble
XGBoost	0.7	0.8	0.69	0.71	False	MaxOutputEnsemble
Random Forest	0.7	0.78	0.63	0.78	False	resnet
Naive Bayes	0.7	0.79	0.65	0.76	False	efficientnet
MLP 1000	0.7	0.81	0.72	0.67	False	efficientnet
MLP 1000 relu 0.001	0.7	0.81	0.72	0.67	False	efficientnet
MLP 300 200 100	0.7	0.79	0.66	0.74	False	efficientnet
MLP 1000 tanh	0.69	0.77	0.62	0.77	False	resnet
MLP 500 500	0.69	0.76	0.6	0.81	False	vgg
MLP 300 200 100	0.69	0.77	0.63	0.76	False	vgg
AdaBoost	0.69	0.77	0.62	0.78	False	vgg

Table 4.1 Results for combined text and image data on ProMapEn dataset.

4.0.3 Results on ProMapCz Dataset

The results of combining text and image data on the ProMapCz dataset are presented in Table 4.2.

Model	F1	Acc	Prec	Rec	Name	Image
MLP 500 500	0.69	0.75	0.58	0.87	False	efficientnet
AdaBoost	0.68	0.76	0.6	0.78	False	MaxOutputEnsemble
MLP 1000 relu adam	0.67	0.71	0.53	0.91	False	efficientnet
MLP 300 200 100	0.67	0.71	0.53	0.91	False	MaxOutputEnsemble
Bagging	0.67	0.76	0.61	0.76	False	MaxOutputEnsemble
MLP 300 200 100	0.67	0.73	0.55	0.84	False	efficientnet
MLP 1000 relu 0.001	0.67	0.73	0.56	0.83	False	MaxOutputEnsemble
MLP 1000	0.67	0.73	0.56	0.83	False	MaxOutputEnsemble
SGD	0.67	0.76	0.6	0.77	True	MaxOutputEnsemble
MLP 1000 tanh	0.66	0.75	0.59	0.76	False	MaxOutputEnsemble
MLP 1000 relu adam	0.66	0.74	0.58	0.78	True	vgg
AdaBoost	0.66	0.74	0.58	0.77	False	efficientnet
MLP 1000 tanh	0.66	0.73	0.56	0.8	False	efficientnet
SVC	0.66	0.74	0.58	0.76	False	MaxOutputEnsemble
MLP 1000 relu adam	0.66	0.74	0.58	0.78	True	MaxOutputEnsemble
Random Forest	0.66	0.75	0.59	0.74	False	efficientnet
Bagging	0.66	0.76	0.62	0.7	True	efficientnet
Logistic Regression	0.66	0.75	0.59	0.74	False	MaxOutputEnsemble
MLP 500 500	0.66	0.72	0.54	0.84	False	vgg
MLP 300 200 100	0.66	0.73	0.56	0.8	False	vgg

Table 4.2 Results for combined text and image data on ProMapCz dataset.

The highest F1 score of 0.69 was achieved using the efficientnet and MLP classifier 1.4.

Here, we can also see that the top image pretrained models are efficientnet and MaxOutputEnsemble.

4.0.4 Comparative Analysis of F1 Scores

After testing on datasets such as ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart, our findings indicate that fine-tuning and employing multimodal approaches significantly enhance accuracy and F1 scores. Specifically, our experiments yielded the following highest F1 scores:

- ProMapEn: 0.73 using all features (text + images) with an MLP classifier 1.4, and 0.72 using only text features with an SVC classifier 1.4.
- ProMapCz: 0.72 with a fine-tuned BERT model.
- Amazon-Google: 0.99 with a fine-tuned BERT model.
- Amazon-Walmart: 0.93 with a fine-tuned BERT model.

To contextualize our results, we compare them with those reported by in ProMap datasets for Product Mapping in e-commerce [6]. Table **??** presents a comparative analysis, showcasing the performance metrics side by side.

Dataset	Our F1 Score	Their F1 Score	Difference
ProMapEn	0.73	0.7	0.03
ProMapCz	0.72	0.77	-0.05
Amazon-Google	0.99	0.99	0.0
Amazon-Walmart	0.93	0.93	0.0

Table 4.3 Comparison of F1 Scores: Our Models vs. reference results from [6].

Table **??** provides a comparative analysis of the F1 scores achieved by our models and those reported by Macková and Pilát [6]. The analysis reveals several key insights:

- **ProMapEn**: Our model achieved an F1 score of 0.73, which is an improvement of 0.03 over the reference results. This improvement can be attributed to the use of multimodal features (text and images) and use of Serialized model for sentence similarity 3.6.1. Also, lets notice that we, also, outperformed the result by 0.02 without using the image data.
- **ProMapCz**: Our model achieved an F1 score of 0.72, which is 0.05 lower than the reference results [6]. This difference may be due to the specific techniques and feature selection methods employed in their approach. Also, let's note that the image data is not used to achieve an F1 score of 0.72.
- **Amazon-Google**: Both our model and the reference results [6] achieved an identical F1 score of 0.99. This consistency suggests that the fine-tuned BERT model is highly effective for this dataset.
- **Amazon-Walmart**: Similarly, both models achieved an F1 score of 0.93, indicating that the methodologies used are robust for this dataset.

Overall, the comparative analysis demonstrates that our models are competitive, achieving higher F1 scores on certain datasets and performing equally well on others.

Conclusion

We ran the experiments by applying deep-learning techniques for Product Mapping. The thesis starts with traditional text classification approaches like TF-IDF and Word2Vec and moves on to using BERT-based architectures for text data. We used Convolutional Neural Networks like VGG16, ResNet50, Inception, and EfficientNet for image processing. We have tested our approaches on ProMapEn, ProMapCz, Amazon-Google, and Amazon-Walmart datasets with the help of machine learning and deep learning approaches **??**. We found that fine-tuning BERT and multimodal approaches improves accuracy and F1 scores. This observation highlights the importance of deep learning in Product Mapping. We got a value of 0.73 for the F1 score with the ProMapEn data using MLP classifiers and 0.72 with text features only using an SVC classifier. For the ProMapCz dataset, the best score was 0.72 using the fine-tuned BERT model. Using a fine-tuned BERT model, we achieved an F1 of 0.99 in the Amazon-Google dataset and 0.93 in the Amazon-Walmart dataset.

Other results are also shown for comparison purposes [6]. More precisely, our F1 score on the ProMapEn dataset has increased for both image and text data and F1 for text data only. While the F1 score was not high in the ProMapCz dataset, it has huge potential to fine-tune the multitask training process when integrated with data from images. The comparative analysis showed that our models are competitive, with even higher F1 on some datasets and equal on others.

This opens many promising avenues for improvement in developing Product Mapping. In this view, it can be important to have big language models to find text and image similarity. Such models as ChatGPT, LLama3, and Gemini would enhance better understanding and precision in matching product descriptions to images. Moreover, category-specific training models will show much better performance in catching the peculiarities of categories. Such an approach may improve the precision of matching products in a particular domain.

Other features such as brand names or other characteristics could also be included to enhance the data processing techniques. This enriched representation would lead to more accurate product mappings. More advanced deep learning methods and architectures could also be explored, such as self-supervised learning, generative models, or ensemble learning, which may bring new insights and improvements to product mapping. This thesis discusses the analysis of the different techniques and their performance on varied datasets, creating a solid ground for future work in Product Mapping.

Bibliography

1. Primpeli, Anna; Peeters, Ralph; Bizer, Christian. The WDC training dataset and gold standard for large-scale product matching. In: *Companion Proceedings of The 2019 World Wide Web Conference (WWW '19)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 381–386. Available from DOI: 10.1145/3308560.3316609.
2. Naumann, F. *Amazon-Walmart dataset* [https://hpi.de/naumann/projects/repeatability/datasets/amazon-walmart-dataset.html]. [N.d.].
3. Rahm, E.; Peukert, D.E.; Saeedi, A.; Nentwig, M. *Benchmark datasets for entity resolution* [https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution]. [N.d.].
4. Rahm, Erhard; Peukert, Eric; Saeedi, Alieh; Nentwig, Markus. *Benchmark datasets for entity resolution* [https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution]. [N.d.]. Accessed: 2024-07-17.
5. Köpcke, Hanna; Thor, Andreas; Rahm, Erhard. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*. 2010, vol. 3, no. 1-2, pp. 484–493. Available from DOI: 10.14778/1920841.1920904.
6. Macková, Kateřina; Pilát, Martin. *ProMap: Datasets for Product Mapping in E-commerce*. 2023. Available from arXiv: 2309.06882 [cs.LG].
7. Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. Available from arXiv: 1810.04805 [cs.CL].
8. Chen, Tianqi; Guestrin, Carlos. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: ACM, 2016, pp. 785–794. KDD '16. ISBN 978-1-4503-4232-2. Available from DOI: 10.1145/2939672.2939785.
9. Cox, David R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*. 1958, vol. 20, no. 2, pp. 215–232.
10. Ho, Tin Kam. Random decision forests. In: *Proceedings of 3rd international conference on document analysis and recognition*. IEEE, 1995, vol. 1, pp. 278–282.
11. Cortes, Corinna; Vapnik, Vladimir. Support-vector networks. *Machine learning*. 1995, vol. 20, no. 3, pp. 273–297.
12. Fix, Evelyn; Hodges, Joseph L. *Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties*. 1951. Report. USAF School of Aviation Medicine, Randolph Field, Texas. Available also from: https://archive.org/details/discriminatory-analysis.

13. Sammut, Claude; Webb, Geoffrey I. (eds.). TF–IDF. In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 986–987. ISBN 978-0-387-30164-8. Available from DOI: 10.1007/978-0-387-30164-8_832.

14. Haykin, Simon. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

15. Schapire, Robert E. Explaining adaboost. In: *Empirical inference*. Springer, 2013, pp. 37–52.

16. Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*. 2001, pp. 1189–1232.

17. Geurts, Pierre; Ernst, Damien; Wehenkel, Louis. Extremely randomized trees. *Machine Learning*. 2006, vol. 63, no. 1, pp. 3–42. Available from DOI: 10.1007/s10994-006-6226-1.

18. Fan, Wei; Zhang, Kun. Bagging. In: *Encyclopedia of Database Systems*. Ed. by LIU, LING; ÖZSU, M. TAMER. Boston, MA: Springer US, 2009, pp. 206–210. ISBN 978-0-387-39940-9. Available from DOI: 10.1007/978-0-387-39940-9_567.

19. Ruder, Sebastian. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. 2016.

20. Simonyan, Karen; Zisserman, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. Available from arXiv: 1409.1556 [cs.CV].

21. Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai; Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

22. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian. *Deep Residual Learning for Image Recognition*. 2015. Available from arXiv: 1512.03385 [cs.CV].

23. Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew. *Going Deeper with Convolutions*. 2014. Available from arXiv: 1409.4842 [cs.CV].

24. Tan, Mingxing; Le, Quoc V. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019. Available from arXiv: 1905.11946 [cs.LG].

25. Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey. *Efficient Estimation of Word Representations in Vector Space*. 2013. Available from arXiv: 1301.3781 [cs.CL].

26. Tracz, Janusz; Wójcik, Piotr Iwo; Jasinska-Kobus, Kalina; Belluzzo, Riccardo; Mroczkowski, Robert; Gawlik, Ireneusz. *Proceedings of Workshop on Natural Language Processing in E-Commerce*. BERT-based similarity learning for product matching. Ed. by Zhao, Huasha; Sondhi, Parikshit; Bach, Nguyen; Hewavitharana, Sanjika; He, Yifan; Si, Luo; Ji, Heng. Barcelona, Spain: Association for Computational Linguistics, 2020. Available also from: https://aclanthology.org/2020.ecomnlp-1.7.

27. Reimers, Nils; Gurevych, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. Available also from: http://arxiv.org/abs/1908.10084.

28. Golestani, Abbas. *ag-nli-DeTS-sentence-similarity-v1* [https://huggingface.co/abbasgolestani/ag-nli-DeTS-sentence-similarity-v1]. 2023. Accessed: 2023-07-12.

List of Figures

List of Tables

List of Abbreviations

- **TF-IDF**: Term Frequency-Inverse Document Frequency
- **Word2Vec**: Word to Vector
- **BERT**: Bidirectional Encoder Representations from Transformers
- **CNN**: Convolutional Neural Network
- **VGG16**: Visual Geometry Group 16
- **ResNet50**: Residual Network 50
- **SVC**: Support Vector Classifier
- **MLP**: Multi-Layer Perceptron
- **ProMapEn**: Product Mapping English dataset
- **ProMapCz**: Product Mapping Czech dataset

A Attachments

A.1 Source Code

- **Code Repository**: `https://gitlab.mff.cuni.cz/teaching/nprg045/mackova/hovsepyan_mkrtich`

A.2 Datasets

- **ProMap, Amazon-Google, and Amazon-Walmart**: `https://github.com/kackamac/Product-Mapping-Datasets`