Semestral project

Adam Vrba

July 2025

1 Introduction

In this semestral project, we compare the gradient weight search algorithm against the evolutionary approach for finding weights of neural networks. In chapter 2 we describe the product datasets on which we will train and test neural networks. In the chapter 3 we briefly describe which algorithms we will use in the experiments. In chapter 4 we describe preprocessing methods and used hyperparameters for the gradient and evolutionary approach. Lastly, for each dataset and preprocessing method, we compare the best networks found by the evolutionary and traditional approaches.

2 ProMap datasets

Product matching is a process of comparing a pair of products, each taken from different e–shops, and determining if they are the same (matching) or if they are different (non-matching). Each product in the entire ProMap dataset contains the name of a product, short and long description, specification, product image, and product price.

Furthermore, we categorize non-matching pairs into two subgroups: close non-match and medium non-match. Close non-match is defined as the product from the target website which is very similar but not the same as the source product. It is supposed to have the same brand, similar name, and similar price. A medium non-match is a product that is different but still has a number of similar features. Products which do not match in these features are defined as distant non-match. For our problem, we decided to use the ProMap dataset which contains data information about products.

Thus ProMap datasets are considered a golden standard for product matching, which were created by Macková and Pilát [7]. With the help of human annotators, they created two new datasets: ProMapCz and ProMapEn, and resampled and edited already existing datasets: Amazon–Walmart [1] and Amazon–Google [5, 8]. In the following subsections, we will discuss the quality and creation of ProMap datasets, and in the last subsection we will discuss how ProMap features were preprocessed into numerical values.

2.1 ProMapCz

ProMapCz contains 1495 unique products from Mall.cz and 706 unique products from Alza.com across 10 different categories: pet supplies, backpacks and bags, hobby and garden, appliances, mobile phones, household supplies, laptops, TVs, headphones, and fridges. From these products, 1495 pairs of products were created which form 509 matching pairs, 456 close non-matching pairs, and 535 medium non-matching pairs.

2.2 ProMapEn

ProMapEn contains 1555 unique products from Walmart.com and 751 unique products from Amazon.com across 10 different categories: pet supplies, backpacks and accessories, patio and garden, kitchen appliances, mobile phones, home improvement, laptops, toys, sports and clothes, health and beauty. From these products, 1555 pairs of products were created which form 509 matching pairs, 509 close non-matching pairs, and 537 medium non-matching pairs.

2.3 Amazon–Walmart

Original Amazon–Walmart datasets [1] were resampled so they would have the same distribution of matching and non-matching as ProMapCz and ProMapEn datasets. Resampled Amazon–Walmart dataset contains 1143 matching and 2000 non-matching product pairs. This resampled dataset contains name of products, short and long descriptions, price of both products, and a binary identifier which tells if it is matching or non-matching.

2.4 Amazon–Google

Even Amazon–Google dataset was resampled from the original dataset [5], so it would have the same distribution of matching and non-matching pairs of products. Amazon–Walmart dataset contains 1143 matching and 2000 non-matching product pairs. This dataset contains the name of products, short and long descriptions, price of both products, and a binary identifier which tells if it is matching or non-matching.

2.5 Dataset preprocessing

In this subsection we will in detail explain how the ProMap datasets were preprocessed into numerical features [7].

2.5.1 Image preprocessing

Each image from the target sites was converted to gray scale, on which the largest object was automatically detected. The found object was then cropped from the image, and after obtaining the largest object, the image was converted to a numerical value using the perceptual hashing method[10]. After obtaining

all numerical values of the images, the values were compared between corresponding product pairs. After converting all images to numerical values, the numerical hash values were compared between a pair of products, and the overall similarity between the images was obtained. Each product image from the source set was compared with all images from the target set. Images that had at least 90% similarity were retained. Finally, all resulting similarities were summed up. We will refer to this feature in the work as hash similarity [7].

2.5.2 Text preprocessing

From the whole text which product contains (such as name of the product, long and short description etc.), useless characters were deleted, such as white space, brackets, etc. Values with units which were not separated by space were separated by space. Then the whole text was separated into single words and lemmatized using the morphological analyzer Ajka [9]. Lastly, the whole text was converted to lower case. This process created a new feature called all_text consisting of the concatenation of the name, long and short descriptions and specifications.

The next step is to compute text similarity To compute the text similarity we convert the preprocessed text into a numerical vector tf.idf. The text similarity between products is a numerical value calculated over the cosine distance of the two vectors v_1 and v_2 as $\frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|}$, where $|v_i|$ is the length of the vector v_i. This process creates three additional features: name_cos, short_description_cos and long_description_cos.

Because textual fields can contain useful information with special meanings such as product ID, brand, units, numbers, and descriptive words.

ID detection is performed by selecting unique words longer than five characters that are not included in vocabularies created based on ParaCrawl [2] corpus. IDs detection was performed on name product name, short description and all text, which gave us features: name_id, short_description_id and all_texts_id.

Brand detection is performed on vocabulary created by scraping all brands on the source and target website. Brands are detected in name, short description and all text giving us features: name_brand, short_description_brand and all_texts_brand.

Numbers detection is based on detecting numbers in the text and searching for units around them. Then numbers which do not have units near them were detected and compared in every feature giving us features: name_number, short_description_numbers, long_description_numbers, specification_text_numbers and all_text_numbers.

Units were determined in a similar way as number detection. A unit is detected if a number is followed by a unit. After detecting units, we calculated similarity on the detected units. Unit detection is also performed on all features. Thus, we get the following features: name_units, short_description_units, long_description_units, specification_text_units and all_text_units.

Descriptive words are a set of the most characterising words for each feature in the product. Descriptive words were experimentally determined as the fiftieth

Table 1: List of All Generated Features.

Feature Name		
name_cos	short_description_descriptives	all_texts_words
name_id	short_description_units	all_brands_list
name_brand	short_description_words	all_numbers_list
name_numbers	short_description_cos	all_ids_list
name_descriptives	short_description_brand	all_units_list
name_units	short_description_id	specification_text_numbers
name_words	short_description_numbers	specification_text_units
all_texts_cos	all_texts_brand	specification_key
all_texts_id	all_texts_numbers	specification_key_value
long_description_numbers	all_texts_units	hash_similarity
long_description_cos	all_text_words	long_description_descriptives
long_description_units	all_texts_descriptives	match

most frequent word in 50% of the document to avoid common words such as conjunctions and prepositions. The detection was performed on the product name, short and long description of the specification, and also on the all text tag. And we get the following tags: name_descriptives, short_description_descriptives, long_description_descriptives, and all_text_descriptives.

The next feature we will describe is the creation of words feature. This feature is determined by the ratio of the same words to different words. All words are from the corresponding features of the two products, namely the product name, short description and all text feature. In total, we get new features: name_words, short_description_words and all_text_words.

After keyword detection, for each detected keyword, a new list was created, and the ratio of matching values in those lists were compared between two products, giving us new features: all_units_list, all_ids_list, all_numbers_list and all_brand_list..

Lastly, we will discuss the preprocessing applied to the product specifications. Numbers followed by units were extracted and preprocessed and were compared between two products to obtain parameters and their similarity as they can specify the product in detail. Because the values can be in different units, all values were transformed to basic unit form without any prefixes. For the corresponding parameter names and their values, the ratio was computed. The ratio between parameters we denote as specification key_matches and the ratio between values we denote as specification_key_value.

2.6 Summary

In summary, ProMap datasets contain 35 features (which are listed in table 1). For our experiments, we will use the preprocessed versions of the ProMap dataset which we have described in this section. The original ProMap datasets as well as the preprocessed ProMap datasets are available on GitHub [6].

3 Weight search methods

As we mentioned in the introduction, we use gradient and evolutionary algorithms to find the weights of neural networks. In this section, we will in detail explain both approaches.

3.1 Gradient approach

For the gradient-based optimization we can use stochastic gradient descent (SGD) or Adam (adaptive moment estimation) [4]. Both of these algorithms are first-order gradient-based optimization algorithms well-suited for stochastic objective functions, which in the context of neural networks refers to the loss function. Given that product mapping involves binary target features, we utilized the binary cross-entropy loss function, defined as:

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

The primary objective of both methods is to minimize this loss function. For the gradient-based approach, we selected Adam over classic stochastic gradient descent (SGD) because Adam converges faster than the basic SGD algorithm.

To ensure a fair comparison with the evolutionary approach, which contains a population of neural networks, we decide to use grid search to identify the optimal hyperparameters for the Adam optimizer. Grid search is a method for finding the best hyperparameters for neural networks. Grid search trains the neural network on all possible combinations of hyperparameters. The specific hyperparameters will be described in detail in section 4.

3.2 Evolutionary approach

For the evolutionary approach, we employ CMA–ES (Covariance Matrix Adaptation Evolution Strategy) [3]. This strategy is an effective optimization algorithm for non-linear problems in continuous domains. It operates by changing parameters of a multivariate normal distribution $\mathcal{N}(\mu, C)$ where $\mu \in \mathbb{R}^n$ is a mean vector and $C \in \mathbb{R}^{n\times n}$ is a symmetric positive definite covariance matrix.

At the start of each generation, individuals are created by sampling from the multivariate normal distribution. After creation, each individual is assigned a fitness score to quantify its performance.

For our weight search problem, we can choose from various metrics to assign each individual's fitness such as accuracy, F1–score, precision, recall, loss function, or any other metric that is used in machine learning. For the experiments, we have chosen F1–score as the primary fitness metric because this metric takes into account unbalanced datasets.

After assigning the fitness to our population, we select the best solutions, and by these selected individuals, we update our covariance matrix and mean vector. This process is repeated until we converge or achieve the maximum number of generations.

4 Experiment results

For each ProMap dataset, we use 3 different preprocessing methods: PCA, LDA, and RAW (no preprocessing was used). After preprocessing the data, we run Gradient and Evolutionary weight search approaches on two layer sizes: small (consisting of 2 hidden layers of size 16 and 8) and large (consisting of 3 hidden layers of size 8, 4, and 2). For each combination of preprocessing method and weight search approach, we will identify the best architecture and compare their F1–score.

4.1 Data preprocessing

In this subsection, we will describe hyperparameters for PCA and LDA preprocessing.

4.1.1 PCA

For PCA preprocessing, we have selected the number of components such that the cumulative explained variance of data is greater than or equal to 95%. Other PCA parameters were kept to default values.

4.1.2 LDA

For LDA preprocessing, we have selected the number of components to be 1. Other LDA parameters were kept to default values.

4.2 Hyperparameters settings

In this subsection, we will discuss hyperparameter settings for the evolutionary and gradient approaches.

4.2.1 Evolutionary approach

For the evolutionary approach, we chose to use the deap library because it provides flexibility and ease of use. As mentioned, we used the CMA-ES algorithm to optimize weights in neural networks. The algorithm was run for 50 generations. We initialized centroid as a vector where each element was 0.1 and the length was equal to the total number of network parameters. The initial standard deviation of the distribution (parameter σ) was set to 0.5. The number of children to produce at each generation (parameter λ) was set to the total number of the network parameters. And lastly, the number of parents to keep from the lambda children (μ) was set to a fourth of λ. The fitness of each individual was evaluated using the F1-score. Other hyperparameters were kept at their default values.

Table 2: Comparison of model performance on training datasets.

Preprocess	Train	Method	F1	Pre	Rec	Acc	Size
lda	google	backpropagation	0.986513	0.996109	0.977099	0.989181	(8, 4, 2)
lda	google	evolutionary	0.984733	0.984733	0.984733	0.987635	(8, 4, 2)
pca	google	evolutionary	0.984615	0.992248	0.977099	0.987635	(8, 4, 2)
pca	google	backpropagation	0.978723	0.992157	0.965649	0.982998	(16, 8)
raw	google	evolutionary	0.986564	0.992278	0.980916	0.989181	(8, 4, 2)
raw	google	backpropagation	0.986460	1.000000	0.973282	0.989181	(8, 4, 2)
lda	promapcz	backpropagation	0.734694	0.734694	0.734694	0.826087	(8, 4, 2)
lda	promapcz	evolutionary	0.734694	0.734694	0.734694	0.826087	(16, 8)
pca	promapcz	evolutionary	0.339869	0.472727	0.265306	0.662207	(16, 8)
pca	promapcz	backpropagation	0.000000	0.000000	0.000000	0.672241	(16, 8)
raw	promapcz	evolutionary	0.719577	0.747253	0.693878	0.822742	(8, 4, 2)
raw	promapcz	backpropagation	0.702703	0.747126	0.663265	0.816054	(16, 8)
lda	promapen	backpropagation	0.598802	0.757576	0.495050	0.784566	(16, 8)
lda	promapen	evolutionary	0.602273	0.706667	0.524752	0.774920	(16, 8)
pca	promapen	backpropagation	0.604938	0.803279	0.485149	0.794212	(16, 8)
pca	promapen	evolutionary	0.568182	0.666667	0.495050	0.755627	(8, 4, 2)
raw	promapen	backpropagation	0.602410	0.769231	0.495050	0.787781	(16, 8)
raw	promapen	evolutionary	0.595506	0.688312	0.524752	0.768489	(8, 4, 2)
lda	walmart	evolutionary	0.919149	0.892562	0.947368	0.939587	(16, 8)
lda	walmart	backpropagation	0.918103	0.902542	0.934211	0.939587	(16, 8)
pca	walmart	backpropagation	0.904348	0.896552	0.912281	0.930048	(8, 4, 2)
pca	walmart	evolutionary	0.902386	0.892704	0.912281	0.928458	(8, 4, 2)
raw	walmart	backpropagation	0.907127	0.893617	0.921053	0.931638	(8, 4, 2)
raw	walmart	evolutionary	0.902386	0.892704	0.912281	0.928458	(8, 4, 2)

4.2.2 Gradient approach

To ensure a fair comparison and identify optimal hyperparameters, we employed a grid search with 5–fold cross–validation for 50 max iterations to determine the best hyperparameters for the gradient approach. This grid explored various activation functions such as relu, tanh, and logistic, parameter α set to 0.0001 and 0.001, and hidden layer sizes set to small (16, 8) and large (8, 4, 2). After identifying the optimal hyperparameters, we retrained the network on the best hyperparameters for 50 iterations.

4.3 Results

In table 2 we compared the best models found by two approaches and on all datasets including the preprocessed dataset via PCA and LDA.

Models trained and validated on the Amazon–Google dataset achieved 97% to 98% F1–score. We observe no significant difference between the evolutionary and gradient approaches, nor among the different preprocessing techniques.

On ProMapCz, the results were different. LDA models with evolutionary and gradient approach achieved 73% F1–score, whereas models trained on raw dataset achieved 70% to 71% F1 score. The worst result was when we used PCA preprocessing technique. Gradient approach achieved 0% F1–score and evolutionary approach achieved only 33% F1–score. This poor performance on both approaches is attributed to the dimensionality reduction performed by PCA, which probably removed too much critical information, making the dataset extremely challenging.

Models trained and validated on every preprocessed ProMapEn dataset achieved 59% to 60% F1 score. Only the evolutionary approach on the PCA preprocessed dataset achieved 56% accuracy.

Lastly, on the Amazon–Walmart dataset, all methods on all types of preprocessing achieved 90% to 91% F1–score.

5 Summary

Results showed that evolutionary algorithms can achieve models which have similar performance as models trained with gradient approaches. Data preprocessing did not improve our model by a significant margin, only the LDA method on ProMapCz datasets achieved around 3% better performance than models trained on the unprocessed dataset.

A significant limitation of evolutionary weight search appeared with larger and deeper neural networks (for example networks with layer size (1000) or even with size (100, 50)), because they often failed to complete even their initial generation within a reasonable time. This leads to the conclusion that while evolutionary weight search can yield good results on small networks, its computational speed is significantly slower compared to gradient–based weight search methods.

References

[1] *Amazon-walmart dataset*. URL: https://hpi.de/naumann/projects/repeatability/datasets/amazon-walmart-dataset.html (visited on 04/02/2025).

[2] Marta Bañón et al. "ParaCrawl: Web-Scale Acquisition of Parallel Corpora". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 4555–4567. DOI: 10.18653/v1/2020.acl-main.417. URL: https://aclanthology.org/2020.acl-main.417/.

[3] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. 2023. arXiv: 1604.00772 [cs.LG]. URL: https://arxiv.org/abs/1604.00772.

[4] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[5] Hanna Köpcke, Andreas Thor, and Erhard Rahm. "Evaluation of entity resolution approaches on real-world match problems". In: *Proc. VLDB Endow.* 3.1–2 (Sept. 2010), pp. 484–493. ISSN: 2150-8097. DOI: 10.14778/1920841.1920904. URL: https://doi.org/10.14778/1920841.1920904.

[6] Kateřina Macková. *Product-Mapping-Datasets*. URL: https://github.com/kackamac/Product-Mapping-Datasets (visited on 04/02/2025).

[7] Kateřina Macková and Martin Pilát. *ProMap: Datasets for Product Mapping in E-commerce*. 2023. arXiv: 2309.06882 [cs.LG]. URL: https://arxiv.org/abs/2309.06882 (visited on 03/13/2025).

[8] Erhard Rahm Dr. Eric Peukert Alieh Saeedi and Markus Nentwig. *Benchmark datasets for entity resolution*. URL: https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution (visited on 04/02/2025).

[9] Radek Sedláček and Pavel Smrž. "A New Czech Morphological Analyser ajka". In: *Text, Speech and Dialogue*. Ed. by Václav Matoušek et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 100–107. ISBN: 978-3-540-44805-1.

[10] Bian Yang, Fan Gu, and Xiamu Niu. "Block Mean Value Based Image Perceptual Hashing". In: *2006 International Conference on Intelligent Information Hiding and Multimedia*. 2006, pp. 167–172. DOI: 10.1109/IIH-MSP.2006.265125.