Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

DL workshop 3 - product search report

Task Description:
Our task is to develop a model that predicts the relevance score for various search terms and product combinations from Home Depot's website. Relevance scores range from 1 to 3, with 1 indicating a low level of relevance and 3 indicating a high level of relevance. Using the test set, the goal is to predict the relevance of each pair. Search terms that have been seen and those that have not been seen are included in the test set.

Dataset Description:
Home Depot's website contains real customer search terms and products. Additionally, each search-product pair is assigned a relevance score. The dataset will have columns representing the search term, the corresponding product, and the relevance score.

Training Data:
Contains **search queries** and **product titles** with a **relevance score**. The relevance score indicates how relevant the search term is to the product, it's a score between 1 to 3, with 3 being highly relevant. **product_uid** is a unique identifier for each product.

Testing Data:
Structured similarly to the training data but used to validate the models built from the training set. It includes relevance scores for evaluation.

Attributes Data:
Provides additional information about products, such as specifications or features highlighted in bullet points.

Descriptions Data:
Contains detailed descriptions of the products.

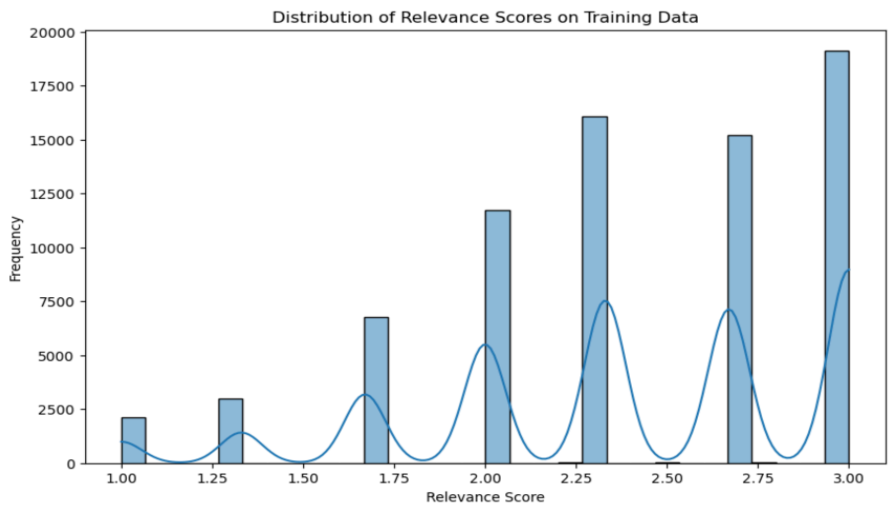
data	number of samples	number of features
training data	74067	5
testing data	166693	5
attribute data	2044803	3
description data	124428	2

Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

<u>Exploratory Data Analysis:</u>

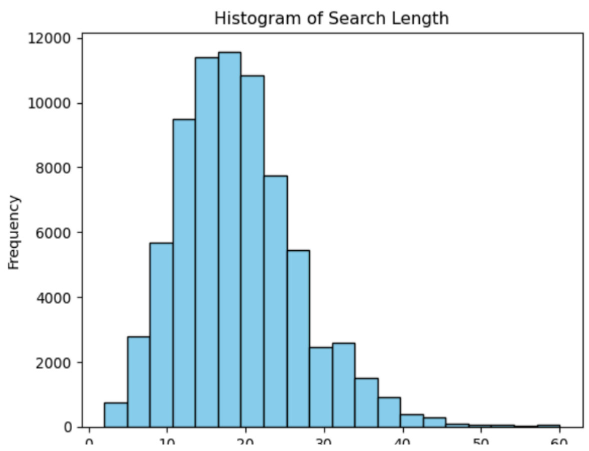
We do EDA to text data and we explore the data and see how the words are distributed, if there is a correlation between the text and the relevance score and more.

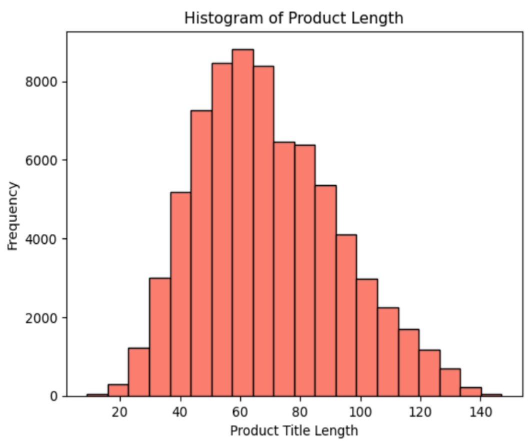
1). relevance score distribution



We can see that most of the relevance scores are higher than two. This indicates that most search terms correlated to the product title.

2). Text length of term search and product title





We can see from the left plot that the search term length has a normal distribution, mean is approximately 26 and there are no search terms longer than 60. Additionally, we can see from the right plot that the product title length has a normal distribution, mean is approximately 60 and there are no search terms longer than 140.

Baseline Naive model:

We implemented a feature extraction pipeline where we transform product titles and search terms into numerical features using CountVectorizer, focusing on character-level details to capture the nuances of language. We then apply these features to train two predictive models:
a RandomForestRegressor, known for its ensemble learning capabilities, and an **XGBoostRegressor**, these models are crucial for predicting how closely a search term matches a product title, these models serve as our baseline to benchmark.

model	**Train RMSE**	**Val RMSE**	**Test RMSE**	**Train MAE**	**Val MAE**	**Test-MAE**
Random Forest	0.1935	0.5066	1.3793	0.1564	0.4130	1.3716
XGBoost	0.5066	0.5179	1.3915	0.3740	0.4228	1.3824

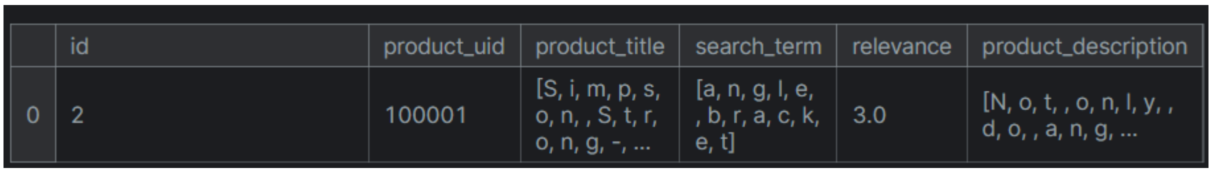
This benchmark score serves as a reference point for evaluating the performance of other models. By comparing the results of the more advanced models, such as those using feature extractors or word-level and character-level processing, it can be determined how well these models perform compared to the benchmark.

Part 1 - Preprocessing Before Siamese network:

Since we are required to build two models, we need to do two different types of data preprocessing.

Character level preprocessing:

This preprocessing step converts the text into sequences of single characters.
First, we used a regular expression to remove any special characters from the input text, leaving only alphabets (both upper and lowercase), digits, and whitespaces.
Then, we convert the string into a list of characters. This is how the data looks like-



	id	product_uid	product_title	search_term	relevance	product_description
0	2	100001	[S, i, m, p, s, o, n, , S, t, r, o, n, g, -, ...	[a, n, g, l, e, , b, r, a, c, k, e, t]	3.0	[N, o, t, , o, n, l, y, , d, o, , a, n, g, ...

The basic unit is a character in character-level processing because of that we indexed the chars to numbers.
It can also help deeply understand words not seen during training since the model can learn to infer meaning from the individual characters more accurately and how they are combined. This can be particularly useful in tasks where the test set contains unseen search terms.

In the next step, we have used numbers to convert our data to numeric representation where each number represents a unique character in the 'product_title', 'search_term' and ‘description’ fields of the dataset.
To ensure all the sequences have the same length, we used padding to ensure the input size was fixed for our model. We choose to balance between losing too much information from longer sequences and having to pad too many shorter sequences. This preprocessing will allow a machine learning model to effectively process these fields as input by providing structured and uniformly dimensioned data.

Word level preprocessing:

We started with text tokenization - ByteLevelBPETokenizer on 'product_title', 'search_term' and ‘description’ fields in the dataset.
Tokenization breaks down a text stream into words, phrases, symbols, or other meaningful elements called tokens. In this case, the tokenization was performed to identify words or character combinations, such as "¾," "90°," or "#SC".
After tokenization we trained word embeddings. We trained a Word2Vec model on the tokens. Word2Vec gives us vector representations of words with semantic meanings. The Word2Vec model was trained on all tokens extracted from 'product_title', 'search_term' and ‘description’.
To ensure all the embeddings have the same length, we used padding to ensure the input size was fixed for our model.
From this representation, we can do sentence vector representation by averaging the word vectors and more.

Part 2 - Build Siamese Network:

Character Level model:

After we transform the textual content into a format suitable for character-level modeling by developing character-level embeddings, we give this embeddings as an input to our siamese model.

The model architecture:

Type	**Description**
LSTM	Long Short-Term Memory layer
Linear	Fully connected linear layer
Sigmoid	Activation function
MSE Loss	Loss function

Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

L1 Loss	Metric

This Siamese architecture is designed for tasks where the relationship between two sequences is important, such as comparing the similarity between a search term and a product title/description.
The network receives two inputs (the 'product_title' and the search term/description), and each one of the inputs is passed through the network.
The LSTM layer captures the temporal dependencies within each sequence as described in the work, the linear layer maps these features to a predicted relevance score, and the sigmoid activation ensures this score falls within a sensible range.
The model uses MSE loss for training, optimizing the similarity predictions, and tracks MAE as an interpretable performance metric.
All the results are recorded using Wandb: link

training parameters:

Logger	Checkpoint Callback	Early Stopping	Accelerator	Max Epochs	dropout	normalization
Logs training progress, metrics, and models	Saves model checkpoints based on validation loss	Prevents overfitting by stopping training early if validation loss doesn't improve	Automatic selection for the best training hardware (GPU)	30	No / Yes (0.4)	No / Yes (BatchNorm)

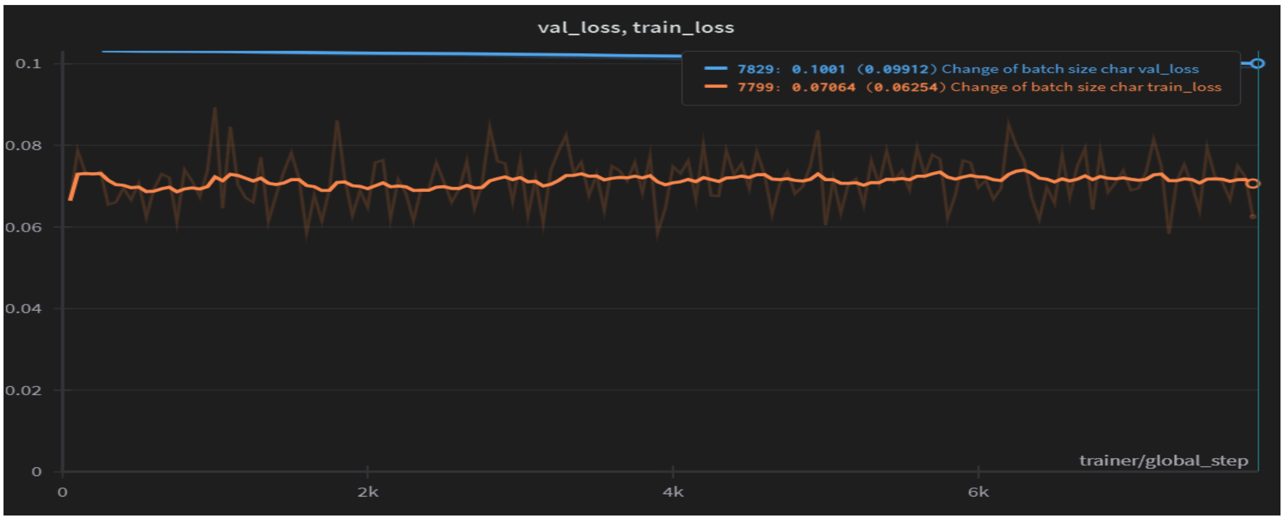
<u>Character Model Results:</u>

	Exp_1	Exp_2	Exp_3	Exp_4	Exp_5
batcn_size	128	128	128	256	128
learning_rate	0.0001	0.0001	0.0001	0.0001	0.01
num_epochs	30	30	30	30	20
train/loss (last)	0.04577	0.0625	0.0706	0.0625	0.06883
test/loss (last)	0.4776	0.479	0.277	0.2722	0.279
dropout	No	0.4	0.4	0.4	0.4
batch_normalization	No	No	Yes	Yes	Yes

Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

The experiment involved training Siamese network models on character sequences for search phrases and item descriptions, using varying batch sizes, dropout, normalization, number of epochs and a constant learning rate of 0.001.
The epochs ranged from 20 to 30, with longer training durations for specific models.
The models exhibited low training losses, indicating a good fit to the training data, with final losses ranging from 0.2722 to 0.479.
However, the test losses were slightly high, suggesting a potential issue of overfitting and limited generalization to unseen test data. Overall, the models showed promising performance, but additional optimization and regularization techniques may be necessary to enhance generalization and mitigate overfitting.

Graph from the best parameterized model (training loss/ validation loss) -



<u>Word Model Results:</u>

	Exp_1	Exp_2	Exp_3	Exp_4	Exp_5
batcn_size	128	128	128	256	128
learning_rate	0.001	0.001	0.001	0.001	0.01
num_epochs	15	15	15	15	20
train/loss (last)	0.03682	0.07505	0.0519	0.0825	0.0939
val/loss (last)	0.06871	0.06914	0.06946	0.0694	0.06884
dropout	No	0.4	0.4	0.4	0.4
batch_normalization	No	No	Yes	Yes	Yes

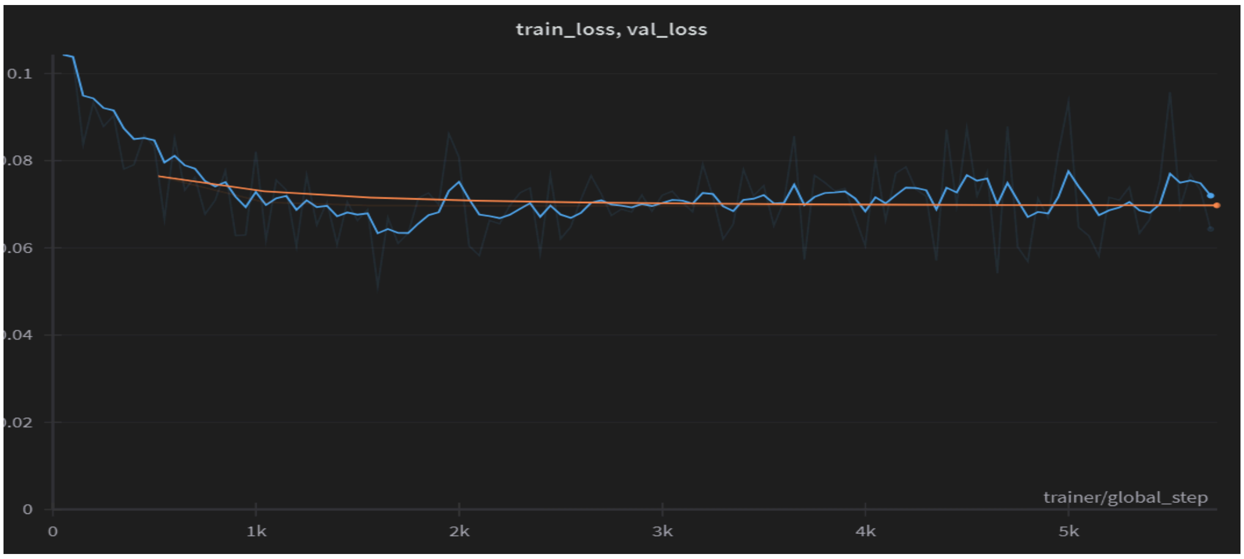
Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

The experiment involved training Siamese network models on word sequences for search phrases and item descriptions, using varying batch sizes, dropout, normalization, number of epochs and a constant learning rate of 0.001.
The epochs ranged from 15 to 20, with longer training durations for specific models. The models exhibited low training losses, indicating a good fit to the training data, with final losses ranging from 0.0519 to 0.0939. The best model is in Exp_3.

While the Siamese network models demonstrated promising performance with low training losses, further optimization, and regularization techniques could be explored to improve generalization and minimize test losses.

An important insight from the gap between the character sequences training loss and test loss compared to word sequences. This suggests that models trained on characters may be more prone to overfitting, diminishing the ability to generalize to new and unseen data. Furthermore, word sequences, providing a more abstract representation, allow for capturing higher-level semantic information and mitigating the risks of overfitting. Thus, opting for word sequences as input enhances the models' generalization and effectiveness in handling novel data.

Graphs from the best parameterized model (training loss/ validation loss) -



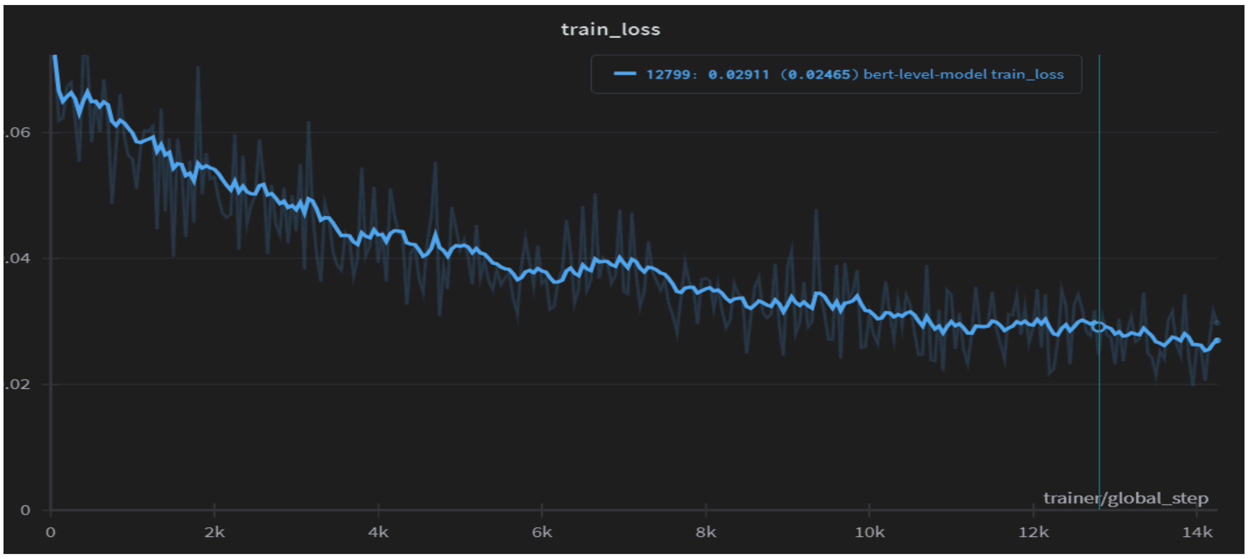
<u>BERT Model Results:</u>

	Exp_1
batcn_size	128
learning_rate	0.0001
num_epochs	1

train/loss (last)	0.1361
val/loss (last)	0.03191

BERT, a powerful pre-trained language model, was incorporated into the architecture to enhance the performance of the Siamese network models.
The model was evaluated using batch sizes 128, number of epochs 1 and a consistent learning rate of 0.001. The model achieved relatively low training losses and test losses, indicating improvement in generalization to unseen data.
Incorporating BERT into the Siamese network architecture holds promise for improving model performance.

Graphs from the best parameterized model (training loss/ testing loss) -



<u>Part 3 – Summary of the results:</u>

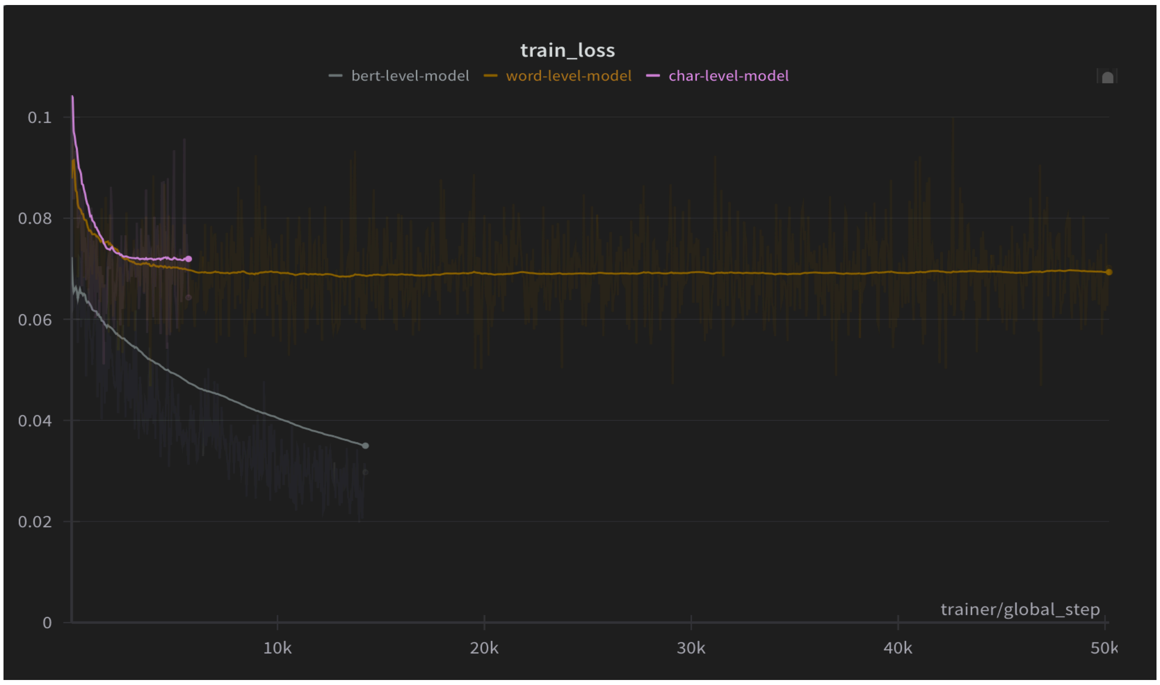
In this part, we tested the performance of four different models:
1. The naive baseline model.
2. The character based model.
3. The word based model using W2Vec
4. The word based model using BERT.

Maxim Katz – 322406604, Danial Hasson -212704597 , Snir Aharon Ezer - 322375783

<u>Results Comparison:</u>

Model type	runtime	Train RMSE	Val-RMSE	Test-RMSE	Train MAE	Val-MAE	Test-MAE
Naïve Random Forest baseline	1,281 sec.	0.1935	0.5071	1.3795	0.1565	0.4133	1.3718
XGBoost baseline	<u>0.6 sec.</u>	0.4609	0.5179	1.3915	0.374	0.4228	1.3824
Character-level LSTM	305.46 sec.	0.2535	0.2639	0.691	0.2	0.2163	0.6911
Word-level LSTM	5,443.44 sec.	0.2386	0.2621	<u>0.6824</u>	0.1856	0.2122	<u>0.6825</u>
BERT-level model	43,106.4 sec.	<u>0.1724</u>	<u>0.1786</u>	-	<u>0.1361</u>	<u>0.1416</u>	-

Graphs that show training loss for all the models types -



The Random Forest naive baseline performed relatively good, and was a solid benchmark for our research.
Our progress is presented in the table above. The character level model trained quickly compared to other models, but still performed well.
The word level model took a lot more time to train, and it improved the results.

Lastly, we tried using a pre-trained BERT model. The training process was extremely slow, and the model trained for one epoch only (that took almost 12 hours) due to lack of resources, but it still got better results overall by a relatively large margin.