Title: Enhancing Search Relevance Prediction in Home Depot Product Search: A Comparative Study of Siamese Networks

Yuri Rusanov, John Zaatra, Majd Azaiza

Institutional Affiliation: Ben-Gurion University of The Negev

Course Number: 372.1.4951

Course Name: A Practical Workshop in Deep Learning

Author Note

This research was conducted as part of the course "A Practical Workshop in Deep Learning" at Ben-Gurion University of The Negev. No grant or funding was received for this project.

GitHub Repository: [[Home-Depot-Item-Relevance-DS-Project](https://github.com/Qehbr/Home-Depot-Item-Relevance-DS-Project)]

Abstract

**Introduction:** This research paper presents a comparative study of Siamese network architectures for predicting search relevance in Home Depot's product search, using data from the Kaggle tournament "Home Depot Product Search Relevance" [‎‎1]. The objective is to assess the efficacy of various models in determining the relevance of search queries to corresponding products and to identify optimal approaches for search relevance prediction.

**Methods:** The study encompasses three key architectural approaches: character-level processing, word-level processing with Word2Vec embeddings, and leveraging BERT-based models. For character-level processing, we have constructed sequences of single characters and fed them into a Siamese network to predict search relevance scores. Similarly, for the word-level approach, word embeddings are generated using Word2Vec and BERT, followed by training Siamese networks to predict relevance scores.

We have successfully done experiments which involve model training and evaluation, comparing the performance of proposed models with benchmark models based on classical machine learning algorithms. Thorough analyses of training time, root mean squared error (RMSE), and mean absolute error (MAE) are conducted to assess the predictive capabilities of each model.

**Results:** Results indicate that the word-based model consistently outperforms the character-based approach and benchmark models in terms of mean absolute error (MAE), achieving the lowest MAE values across all models evaluated. However, when considering root mean squared error (RMSE), the XGBoost model pretrained on word-based embeddings emerges as the top performer, surpassing both the word-based Siamese network and benchmark models in this metric.

In conclusion, the study highlights the effectiveness of Siamese networks, particularly at the word level, in predicting search relevance for Home Depot's product search. The findings underscore the importance of data preprocessing and model selection in optimizing search relevance prediction, providing valuable insights for improving search experiences and product discovery in e-commerce platforms.

Keywords: Siamese networks, search relevance prediction, Home Depot, word embeddings, BERT, machine learning.

Introduction

In an era marked by the ubiquity of e-commerce platforms, shoppers rely on Home Depot’s product authority to find and buy the latest products and to get timely solutions to their home improvement needs. From installing a new ceiling fan to remodeling an entire kitchen, with the click of a mouse or tap of the screen, customers expect the correct results to their queries – quickly. Speed, accuracy and delivering a frictionless customer experience are essential. Consequently, the task of predicting search relevance has become a focal point for companies seeking to optimize product discovery and increase customer satisfaction.

The Kaggle competition "Home Depot Product Search Relevance" serves as a platform for data scientists and machine learning practitioners to tackle this challenge head-on. By providing a dataset comprising search queries, product descriptions, and relevance scores, the competition invites participants to develop predictive models capable of accurately assessing the relevance of search queries to corresponding products.

This research is based on a project that aims to contribute to the advancement of search relevance prediction in e-commerce settings, this research paper presents a comparative study of Siamese network architectures within the context of Home Depot's product search. Siamese networks, known for their effectiveness in learning similarity metrics, offer a promising approach for modeling the relationship between search queries and product descriptions.

The primary objective of this study is to evaluate the efficacy of various Siamese network architectures in predicting search relevance for Home Depot's product search. Specifically, we explore three key architectural approaches: character-level processing, word-level processing with Word2Vec embeddings, leveraging BERT-based models, and a two-layer LSTM architecture. Through extensive experimentation and evaluation, we aim to identify optimal approaches for search relevance prediction and provide insights into the factors influencing predictive performance. However, it's worth noting that a two-layer LSTM architecture were tested but it did not yield satisfactory results compared to other models, warranting further investigation into its limitations and potential improvements.

The structure of this paper is as follows: Section 1 highlights the novel contributions and advancements introduced by our research in the field of search relevance prediction. We discuss the unique approaches explored, the insights gained from the experiments, and how they advance our understanding of search relevance prediction. In Section 2, we provide a comprehensive review of relevant literature in the field of search relevance prediction and Siamese networks. Section 3 outlines the methodology employed in our study, covering data preprocessing, model architectures, and evaluation metrics. Section 4 presents the results of our experiments and discusses key findings. Finally, Section 5 offers concluding remarks and outlines avenues for future research.

# Section 1

**In this section, we discuss the novel contributions and advancements introduced by our research:**

**Innovative Architectural Approaches:** Our study evaluates multiple architectural approaches for search relevance prediction, including character-level processing, word-level processing with Word2Vec embeddings, and leveraging BERT-based models. Additionally, we explore the use of a two-layer LSTM architecture, which, although not as effective as other models, provides insights into its potential applicability in similar contexts.

**Evaluation Metrics:** We employ a range of evaluation metrics, including root mean squared error (RMSE) and mean absolute error (MAE), to assess the predictive capabilities of each model. This comprehensive evaluation allows us to quantify the performance of different architectural approaches and provides a basis for comparing their effectiveness.

**Insights and Practical Implications:** Our research not only contributes to advancing the theoretical understanding of search relevance prediction but also offers practical insights with direct implications for e-commerce platforms. By identifying the most effective architectural approaches and highlighting their strengths and weaknesses, we provide valuable guidance for improving search relevance and enhancing the overall user experience in online shopping environments.

**Potential Impact:** The findings of our study have the potential to impact various stakeholders in the e-commerce ecosystem, including retailers, developers, and consumers. By optimizing search relevance prediction models, retailers can improve product discovery, increase conversion rates, and enhance customer satisfaction. Developers can leverage our insights to design more efficient and effective search algorithms, while consumers can benefit from a more streamlined and personalized shopping experience.

**Section 2**

**Introduction to Literature Review: an overview of Siamese Networks**

**Literature Review**: In recent years, deep learning approaches, particularly Siamese neural networks, have gained significant attention for their effectiveness in addressing challenging machine learning tasks such as one-shot image recognition. The seminal work by Koch et al. (2015) [‎2] introduced the concept of Siamese networks and demonstrated their capability in one-shot learning scenarios, specifically for image recognition tasks.

Koch et al. Proposed a unique architecture together with dual networks with shared weights, termed Siamese networks, to tackle the one-shot getting to know trouble. By training those networks to discriminate among pairs of images from the equal or distinct lessons, they were capable of analyze effective image representations that might generalize to apprehend new classes from just one example, without the want for significant retraining. This approach changed into mainly powerful for duties which include one-shot photograph category, in which traditional machine studying strategies regularly warfare due to restrained labeled records consistent with class.

The architecture employed with the aid of Koch et al. Applied deep convolutional neural networks (CNNs), leveraging the hierarchical functions discovered with the aid of convolutional filters to capture invariant characteristics useful for one-shot classification. The networks have been educated on a large dataset of characters from various alphabets, along with the Omniglot dataset (contains examples from 50 alphabets ranging from well-established international languages like Latin and Korean to lesser-known local dialects), the usage of a verification loss characteristic that assessed whether pairs of pics belonged to the equal class. Through tremendous experimentation, the authors achieved staggering consequences, with their convolutional Siamese network achieving a superb 92% one-shot classification accuracy at the Omniglot evaluation set.

The achievement of Siamese neural networks in one-shot photo recognition tasks has paved the way for further research and exploration within the area of few-shot getting to know. Subsequent studies have constructed upon the rules laid by way of Koch et al., exploring versions of the Siamese structure, novel loss functions, and information augmentation techniques to enhance performance and enlarge the applicability of these models to various domains.

**Objective of the Study:** In the context of our research, understanding the principles and improvements added by using Koch et al. And related works in the discipline of Siamese networks presents valuable insights and thought for designing powerful architectures for seek relevance prediction in e-commerce structures. By leveraging the principles of deep learning and few-shot learning, we aim to develop robust models capable of accurately predicting search relevance with limited labeled data, thereby enhancing the overall user experience in online shopping environments.

**Conclusion:** The literature review has provided an overview of Siamese neural networks and their application in addressing one-shot learning tasks, with a focus on image recognition. The seminal work by Koch et al. demonstrated the effectiveness of Siamese networks in learning powerful image representations capable of generalizing to new classes from just one example.

Furthermore, the literature review emphasized the ongoing research efforts aimed at improving Siamese network architectures, exploring novel loss functions, and enhancing the applicability of these models to diverse domains beyond image recognition.

In the context of our research, the literature review underscores the importance of understanding Siamese networks and their principles for designing effective architectures for search relevance prediction in e-commerce platforms. By building upon the foundations laid by previous works and addressing the identified gaps, we aim to advance the current understanding of search relevance prediction and contribute to the development of robust models capable of enhancing the user experience in online shopping environments.

**Section 3**

**Overview of Methodology:** In this study, we adopt a comprehensive methodology for predicting search relevance in Home Depot's product search. The methodology encompasses three main phases: data preprocessing, model development, and evaluation.

**Data Preprocessing:** Before training the models, we preprocess the dataset to ensure its quality and suitability for the task. This preprocessing involves several steps, including data cleaning to remove any inconsistencies or noise, tokenization to convert text data into manageable units, and normalization to standardize the data and make it more amenable to modeling. Additionally, we apply specific preprocessing techniques tailored to each model architecture to optimize performance (further information’s will be provided later).

**Model Architectures:** We explore multiple architectural approaches for search relevance prediction, including character-level processing, word-level processing with Word2Vec embeddings, BERT-based models, and a two-layer LSTM architecture. Each model architecture is designed to leverage different aspects of the input data and extract relevant features for predicting search relevance scores.

* **Character-level Model:** The character-level model employs a Siamese network architecture to process sequences of single characters extracted from the search queries and product titles. The Siamese network consists of parallel branches with shared weights, allowing it to learn discriminative features from character-level inputs. Here's a breakdown of the layers used in this model:

**Input Layer**: The input to the model consists of sequences of single characters encoded as integers. These sequences are represented as tensors and fed into the model.

**Embedding Layer:** The input sequences are passed through an embedding layer, which converts the integer-encoded characters into dense embeddings of a fixed size. This embedding layer helps in capturing semantic relationships between characters and provides continuous representations for the input data.

**Dropout Layer:** A dropout layer with a dropout probability of 0.3 is applied after the embedding layer to prevent overfitting by randomly dropping out units during training.

**Bidirectional LSTM Layer:** The embedded sequences are then processed by a bidirectional LSTM (Long Short-Term Memory) layer [‎3]. The LSTM layer consists of hidden units responsible for capturing temporal dependencies within the character sequences. The bidirectional nature of the LSTM allows it to capture information from both past and future contexts.

**Global Average Pooling:** After the LSTM layer, a global average pooling operation is applied to the output sequences to aggregate information across all time steps. This pooling operation computes the average value of each feature dimension across the entire sequence.

**Fully Connected Layer:** The output of the global average pooling operation is then passed through a fully connected (dense) layer. This layer performs a linear transformation on the input data and applies a ReLU activation function to introduce non-linearity.

**Batch Normalization:** Batch normalization is applied to the output of the fully connected layer to improve training stability and speed up convergence. It normalizes the activations of each layer across the mini-batch and scales and shifts them using learnable parameters.

**Distance Calculation:** The outputs from both branches of the Siamese network are combined to calculate the Euclidean distance between them. The distance is then passed through a ReLU activation function to ensure non-negativity.

**Sigmoid Activation:** Finally, a sigmoid activation function is applied to the distance value to squash it into the range [0, 1], representing the predicted relevance score between the search query and the product title.

**Train Parameters:**

In this implementation, the character-level Siamese model is configured with specific hyperparameters: the vocabulary size, including an additional padding index (0), the embedding dimension set to 64 to capture intricate character relationships, a hidden dimension of 128 within the LSTM units to potentially capture more complex patterns, a learning rate of 0.001 to control optimization step size while minimizing instability, a batch size of 64 for processing samples in each training iteration, and a total of 50 epochs for training iterations to balance learning from the data without risking overfitting.

This character-level Siamese LSTM model is trained using pairs of input sequences along with their corresponding relevance labels. During training, the model learns to minimize the distance between relevant pairs and maximize the distance between irrelevant pairs, effectively learning to predict search relevance based on character-level information. (several hyperparameters were tested, and the selected ones were found to yield the best performance during experimentation)

For visual assistance you can refer to the Siamese network structure here: [*Figure 1*]

**Benchmark models:** In addition to the implementation of the character-level Siamese model, we also evaluated several benchmark machine learning algorithms for comparison. The implemented algorithms include Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Decision Tree Regressor, Random Forest Regressor, and Gradient Boosting Regressor. These algorithms were trained and tested on the dataset using standard regression techniques. For each algorithm, we recorded various performance metrics, including root mean squared error (RMSE) and mean absolute error (MAE), on both the validation and test sets. This allowed us to assess the predictive performance of the character-level Siamese model in comparison to traditional machine learning approaches.

* **Word-level Model with Word2Vec Embeddings:** The architecture of the word-level Siamese LSTM model follows the same structure as the character-based model, with the main difference lying in the embedding layer. In this model, the input word sequences are passed through an embedding layer initialized with pre-trained Word2Vec embeddings. Word2Vec embeddings offer dense vector representations for words based on their contextual usage, enhancing the model's ability to capture semantic similarities between words. The remaining components of the model, including the dropout layer, bidirectional LSTM layer, global average pooling, fully connected layer, batch normalization, distance calculation, and sigmoid activation, remain consistent with the character-based model. This consistent structure ensures the model's ability to learn discriminative features from word-level inputs and make predictions based on the relevance between search queries and product titles.

**Train Parameters:** The word-level Siamese model is configured with specific hyperparameters: the embedding dimension, which is set to the size of the pre-trained Word2Vec embeddings, hidden dimension of 64 within the LSTM units, learning rate of 0.001 for optimization step size, batch size of 64 for processing samples in each training iteration, and a total of 100 epochs for training iterations.

This word-level Siamese LSTM model is trained using pairs of input sequences along with their corresponding relevance labels. During training, the model learns to minimize the distance between relevant pairs and maximize the distance between irrelevant pairs, effectively learning to predict search relevance based on word-level information. (several hyperparameters were tested, and the selected ones were found to yield the best performance during experimentation)

**Benchmark models:** In addition to the implementation of the word-level Siamese model, we also evaluated several benchmark machine learning algorithms for comparison. The implemented algorithms include Linear Regression, Random Forest, Gradient Boosting Regressor, and XGBoost. These algorithms were trained and tested on the dataset using standard regression techniques same as before.

* **BERT-based Models:** The Bart-level Siamese LSTM model utilizes the Bart-base architecture from Facebook [‎4]. Output vectors of size 768 are generated by the Bart model, which also comes with its own tokenizer. Unlike the character-level and word-level models, the Bart-level model doesn't employ an LSTM layer. Instead, it directly uses the output vectors obtained from the Bart model, which already capture the semantics of the text.

The Siamese network architecture consists of fully connected layers that process the Bart embeddings. These fully connected layers allow the model to learn and capture relevant features from the Bart embeddings. The model utilizes ReLU activation functions between the fully connected layers to introduce non-linearity.

The final output of the model is obtained by calculating the Euclidean distance between the outputs from both branches of the Siamese network. The distance is then passed through a ReLU activation function to ensure non-negativity and is normalized using batch normalization. Finally, a sigmoid activation function is applied to squash the distance value into the range [0, 1], representing the predicted relevance score between the search query and the product title.

For visual assistance you can refer to the Siamese network structure here: [*Figure 2*]

**Train Parameters:** The Bart-level Siamese model is configured with specific hyperparameters, including:

Embedding dimension of 768, which matches the size of the Bart embeddings.

Hidden dimensions for the fully connected layers: 512, 256, and 128.

Learning rate of 0.0001 for optimization step size.

Batch size of 64 for processing samples in each training iteration.

Total of 100 epochs for training iterations.

The model is trained using pairs of input sequences along with their corresponding relevance labels, aiming to minimize the distance between relevant pairs and maximize the distance between irrelevant pairs. (several hyperparameters were tested, and the selected ones were found to yield the best performance during experimentation)

**Benchmark Models:** Similar to the character-level and word-level models, the Bart-level Siamese model is evaluated against several benchmark machine learning algorithms for comparison. The implemented algorithms include Linear Regression, Random Forest, Gradient Boosting Regressor, and XGBoost.

* **Two-layer LSTM:** The DoubleLSTMSiameseLSTM model consists of two LSTM layers, one for processing search terms and the other for processing item descriptions. Each LSTM layer operates independently and processes its respective input sequence. The output of each LSTM layer is then fed into a fully connected layer, followed by a global average pooling operation to aggregate information across all time steps.

The Siamese network architecture ensures that both LSTM branches share parameters up to the fully connected layers, allowing them to learn and capture relevant features from the input sequences. However, unlike traditional Siamese networks where both branches are identical, in this model, the branches contain different LSTM layers tailored for processing search terms and descriptions separately.

The final output of the model is obtained by calculating the Euclidean distance between the outputs from both branches of the Siamese network. The distance is then passed through a ReLU activation function to ensure non-negativity and is normalized using batch normalization. Finally, a sigmoid activation function is applied to squash the distance value into the range [0, 1], representing the predicted relevance score between the search query and the product title.

**Training Procedure:** The model is trained using pairs of input sequences along with their corresponding relevance labels. The training process involves backpropagation to adjust the model parameters, aiming to minimize the mean squared error (MSE) loss between the predicted relevance scores and the ground truth labels. The Adam optimizer is used with a learning rate of 0.001.

**Limitations and Conclusion:** Despite the efforts to design a more complex model with two-layer LSTMs tailored for processing search queries and descriptions separately, the experiment yielded unsatisfactory results. Further analysis and testing of benchmark models were not conducted beyond this point due to the model's poor performance.

**Evaluation Metrics:** The performance of the models is assessed using several evaluation metrics to measure their ability to predict relevance scores between search queries and product descriptions accurately. The primary evaluation metrics utilized in this context include:

* **Root Mean Squared Error (RMSE):**

RMSE is a commonly used metric to evaluate the accuracy of regression models. It measures the average magnitude of the errors between predicted and actual values. Mathematically, RMSE is calculated as the square root of the mean of the squared differences between predicted and actual values. Lower RMSE values indicate better model performance, as they reflect smaller errors between predicted and actual relevance scores.

* **Mean Absolute Error (MAE):**

MAE is another regression metric that measures the average absolute differences between predicted and actual values. Unlike RMSE, MAE does not penalize large errors as heavily, making it more robust to outliers. MAE is calculated as the mean of the absolute differences between predicted and actual values. Similar to RMSE, lower MAE values indicate better model performance, reflecting smaller discrepancies between predicted and actual relevance scores.

These evaluation metrics provide insights into the effectiveness of the models in accurately predicting relevance scores. By analyzing RMSE and MAE values on both training and validation sets, we can assess the generalization performance of the models and identify any potential issues such as overfitting or underfitting.

During model training, the goal is to minimize both RMSE and MAE by optimizing the model parameters through techniques such as backpropagation and gradient descent. The selected hyperparameters and model architecture are adjusted based on the performance metrics obtained during training and validation.

**Experimental Setup and Computational resources:**

**Dataset Splitting:**

* The dataset was split into three subsets: training, validation, and test sets.
* The training set was divided into 80% for training and 20% for validation. This splitting ratio ensures that a substantial portion of the data is used for training while still allocating a separate portion for validation to monitor model performance and prevent overfitting.
* The validation set was utilized to tune hyperparameters, assess model performance during training, and prevent overfitting.
* The test set remained unseen during model development and was reserved for final evaluation to gauge the generalization performance of the trained models.

**Hardware Specifications and Software Frameworks/Libraries:**

* The models were trained and evaluated using GPU (Graphics Processing Unit) acceleration to expedite computation.
* We used PyTorch as it’s a popular deep learning framework, for utilizing the model development, training, and evaluation.
* Additional libraries and tools within the PyTorch ecosystem may have been employed for tasks such as data loading and preprocessing using tensors.

**Further Analysis:** In the evaluation of the models' performance, Train/Validation graphs depicting the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) were generated for the best experiment conducted at each level. These graphs provide valuable insights into the training process and the models' ability to generalize to unseen data.

**Section 4**

**This section presents the results obtained from the experiments conducted. It includes quantitative data such as training time, RMSE, MAE, and any other relevant metrics. Additionally, it discusses key findings, trends, and insights observed from the experiments.**

In this section, we will briefly explain the results for each level, and at the end, we will compare between the levels by summarizing the best results on each level, and to further discuss key finding observed from the final best results of each level.

**Char-level:**

**Benchmark Model Performance Without Stop Words:**

Here is the table presenting the performance metrics of various benchmark models trained without stop words. These models were evaluated based on their root mean squared error (RMSE) and mean absolute error (MAE) on both training and testing datasets. Table: [Table 1]

**Benchmark Model Performance with Stop Words:**

Here is the table illustrating the performance metrics of different benchmark models trained with stop words included. The evaluation metrics include RMSE and MAE computed on the training and testing datasets. Table: [Table 2]

**Siamese Model Performance (Character-level):**

Here is the table displaying the performance metrics of the Siamese model trained at the character-level. The model's training time, along with its RMSE and MAE scores on the training, validation, and testing datasets, are presented. Table: [Table 3]

Train/Validation graphs of RMSE/MAE of best experiment: RMSE [*Figure 3*], MAE[*Figure 4*]

The graphs indicate a promising fit of the Siamese model. In both graphs, the validation loss is consistently higher than the training loss across multiple epochs, indicating a good fit of the model. Additionally, the lines representing RMSE and MAE exhibit a balanced flow, suggesting a robust performance of the model.

**Summary:** The character-level model training was characterized by lengthy training times, particularly for the Siamese model. However, despite the computational cost, the Siamese model did not outperform any of the benchmark models in terms of RMSE and MAE. The Train/Validation graphs for the Siamese model demonstrated a promising fit, with validation losses consistently higher than training losses, indicating good model performance.

**Word-level/Two-layer LSTM:**

**Benchmark Models:**

Here is the table summarizing the performance of various benchmark models on the word-level dataset: [Table 4]

The benchmark models, including Random Forest, Gradient Boosting Regressor, Linear Regression, and XGBoost, were evaluated on the word-level dataset. Among these models, XGBoost exhibited exceptional performance, particularly in terms of RMSE, surpassing other models. This suggests the effectiveness of gradient boosting techniques in capturing complex patterns within the data.

**Siamese Model:**

Below are the results obtained from various experiments conducted using Siamese models on the word-level dataset: [Table 5]

The Siamese models were experimented with different configurations, including variations in preprocessing, word embeddings, and dropout rates. While the Siamese models showed promising performance, with RMSE values competitive with benchmark models, they did not consistently outperform them. However, it is noteworthy that the Siamese models exhibited superior performance in terms of Mean Absolute Error (MAE) compared to all benchmark models, including XGBoost, indicating their effectiveness in capturing the relevance between search queries and product descriptions. Despite this, XGBoost, a traditional machine learning algorithm, still demonstrated superior performance in terms of RMSE.

**Two-layer LSTM Model:**

Results of the two-layer LSTM model are as follows: [Table 6]

The two-layer LSTM model, designed to process search queries and descriptions separately, did not achieve satisfactory results compared to benchmark models. Despite the complexity of the architecture and training process, the model's performance in terms of RMSE and MAE did not surpass that of simpler machine learning algorithms.

Train/Validation graphs of RMSE/MAE of best experiment: RMSE [*Figure 5*], MAE[*Figure 6*]

**Notes on Graphs:** The Train/Validation graphs depicted the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for the best experiments conducted at each level. In these graphs, the lines corresponding to validation loss and training loss showed a close proximity, with the validation loss consistently higher than the training loss over the course of epochs. This trend signifies a good fit of the model to the data, and even better than before.

**Important Results and key findings:**

* Word-based models outperformed all previous models, suggesting an improved ability to capture semantic information.
* XGBoost exhibited exceptional performance, surpassing other models in terms of RMSE.
* The Siamese models demonstrated promising results, with the validation losses consistently higher than the training losses, indicating effective learning.
* In the two-layer LSTM model, despite the separate processing of search queries and descriptions, the performance did not meet expectations, suggesting the need for further optimization or exploration of alternative architectures.

**Bert-level:**

**Benchmark Models:**

Here is the table summarizing the performance of various benchmark models on the BERT-level dataset: [Table 7]

The benchmark models, including Random Forest (RF-100), Gradient Boosting Regressor, Linear Regression, and XGBoost, were evaluated on the BERT-level dataset. Among these models, XGBoost exhibited exceptional performance, particularly in terms of RMSE. While XGBoost showed promising results, it's important to note that it did not surpass its performance on the word-level dataset.

**Siamese Model of BERT:**

Below are the results obtained from various experiments conducted using Siamese models on the BERT-level dataset: [Table 8]

The Siamese models were experimented with different configurations, including variations in BERT embeddings dimensions and learning rates. The models showed promising performance, with competitive RMSE values compared to benchmark models. However, they did not consistently outperform them. Notably, the Siamese model with BERT embeddings of dimension (512, 256, 128) and adjusted learning rate achieved the best RMSE performance, although it still fell short of XGBoost. Despite this, the Siamese model demonstrated superior performance in terms of MAE compared to all benchmark models, indicating its effectiveness in capturing the relevance between search queries and product descriptions.

Train/Validation Graphs of RMSE/MAE for Siamese Model of BERT: RMSE [Figure 7], MAE [Figure 8]

**Notes on Graphs:** Both graphs show a promising fit of the model, with the validation loss consistently higher than the training loss across multiple epochs. Additionally, the lines representing RMSE and MAE exhibit a balanced flow, indicating robust performance of the model.

**Important Notes:**

* The BERT-based model could not exceed the word-level model in terms of MAE.
* Classical machine learning algorithms did not perform well on BERT embeddings, highlighting the effectiveness of more advanced techniques like XGBoost and Siamese networks.

In future improvements, fine-tuning BERT models with domain-specific data and optimizing hyperparameters could potentially enhance their performance on the given task.

In this table [Table 9], you can see the best results we have got from all the levels in this project, from the table we can conclude and summarize the following.

**Summary and Conclusion**

Section 4 provides a comprehensive overview of the experiments conducted across different levels of text representation, including character-level, word-level, and BERT-level, to predict the relevance between search queries and product descriptions. Here's a summary of the key findings and insights obtained from each level:

**Character-Level Analysis:** Benchmark models trained without stop words showcased competitive performance in terms of RMSE and MAE.

However, despite the promising fit observed in the Train/Validation graphs, particularly for the Siamese model, it did not outperform benchmark models in terms of RMSE and MAE.

**Word-Level Analysis:** Word-based models demonstrated superior performance compared to character-level models, emphasizing their enhanced ability to capture semantic information.

XGBoost pretrained on word embeddings emerged as the top performer in terms of RMSE, while Siamese models excelled in MAE, indicating their effectiveness in capturing relevance.

BERT-Level Analysis:

**BERT-level:** Benchmark models trained on BERT embeddings, particularly XGBoost, exhibited exceptional performance in terms of RMSE.

Siamese models leveraging BERT embeddings showed promising results but did not consistently outperform benchmark models. However, they demonstrated superior performance in MAE, highlighting their potential for capturing semantic relevance.

**Overall Insights:**

XGBoost pretrained on word embeddings emerged as the best performer in terms of RMSE, while Siamese models at the word-level outperformed all others in terms of MAE.

Classical machine learning algorithms, such as Random Forest and Gradient Boosting, demonstrated competitive performance but were outperformed by XGBoost across various levels.

Despite the promising performance of Siamese models, there is a need for further optimization, especially in architectures like two-layer LSTM, to achieve satisfactory results comparable to simpler machine learning algorithms.

**Future Directions:**

* Fine-tuning BERT models with domain-specific data and optimizing hyperparameters could enhance their performance.
* Exploring alternative architectures and preprocessing techniques may further improve model performance, especially in character-level and BERT-level models.

**Section 5**

**Conclusion and Discussion:**

The present study contributes significantly to the field of search relevance prediction by exploring novel approaches, gaining valuable insights from experiments, and advancing our understanding of this critical area. The project presented several challenges, particularly in data preprocessing. Our meticulous preprocessing approach tailored to the task likely played a pivotal role in achieving the best results with the word-based model. This underscores the importance of data preprocessing in enhancing model performance.

While BERT is a powerful and complex model, its performance in our study was not superior to our model. One possible explanation is that BERT was trained on diverse text data, unlike our model, which focused specifically on Home Depot items. This suggests that the domain-specific nature of our data and preprocessing techniques may have contributed to the superior performance of our model compared to BERT.

Additionally, it is noteworthy that training the BERT model was considerably faster compared to training our model from scratch. This highlights the trade-off between model quality and training time, which is an essential consideration in practical applications.

The insights gained from this study not only advance our understanding of search relevance prediction but also have practical implications for e-commerce platforms. By employing effective preprocessing techniques and leveraging domain-specific data, e-commerce platforms can enhance search relevance prediction, ultimately improving the user experience and increasing customer satisfaction.

Overall, this study underscores the importance of tailored preprocessing methods and domain-specific modeling approaches in achieving optimal performance in search relevance prediction tasks. It provides valuable insights that can inform future research and guide the development of more effective models for enhancing search relevance in e-commerce settings.

Nomenclature of Abbreviations:  
RF Random Forest

GBR Gradient Boosting Regressor

LR Linear Regression

XGBoost eXtreme Gradient Boosting

LSTM Long Short-Term Memory

MAE Mean Absolute Error

RMSE Root Mean Squared Error

BERT Bidirectional Encoder Representations from Transformers

**Conflict of interest**

Every one of the authors unequivocally state that they have no conflict of interest, regarding all facts, equipment, and procedures involved in this work.

References

1. Kaggle tournament "Home Depot Product Search Relevance". [URL](https://www.kaggle.com/c/home-depot-product-search-relevance/data)
2. Koch, et al. (2015). Siamese Neural Networks for One-shot Image Recognition. [URL](https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf)
3. Long short-term memory – wiki. [URL](https://en.wikipedia.org/wiki/Long_short-term_memory)
4. Facebook – bart face. [URL](https://huggingface.co/facebook/bart-base)

Tables

1. Table 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| With no stop words | Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE |
| Ridge Regression | 0.6924s | 0.5323760 | 0.5308513 | 0.5341565 | 0.4356490 | 0.4333457 | 0.4367494 |
| Lasso Regression | 0.1152s | 0.5336816 | 0.5317526 | 0.5346950 | 0.4374175 | 0.4346863 | 0.4378313 |
| Elastic Net | 0.3929s | 0.5334555 | 0.5316015 | 0.5346117 | 0.4370876 | 0.4344363 | 0.4376308 |
| Decision Tree Regressor | 0.4766s | 0.5315214 | 0.5314483 | 0.5352747 | 0.4355471 | 0.4342663 | 0.4382568 |
| Random Forest Regressor | 34.4749s | 0.5303662 | 0.5302874 | 0.5340345 | 0.4343083 | 0.4330441 | 0.4367363 |
| Gradient Boosting Regressor | 45.2196s | 0.5160257 | 0.5281616 | 0.5338571 | 0.4233715 | 0.4318538 | 0.4371989 |

1. Table 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| With stop words | Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE |
| Ridge Regression | 0.6682s | 0.5325074 | 0.5310246 | 0.5341862 | 0.4358063 | 0.4335874 | 0.4368102 |
| Lasso Regression | 0.1578s | 0.5337974 | 0.5318756 | 0.5347833 | 0.4375612 | 0.4348462 | 0.4379387 |
| Elastic Net | 0.4494s | 0.5336395 | 0.5317419 | 0.5346604 | 0.4373523 | 0.4346742 | 0.4377586 |
| Decision Tree Regressor | 0.4508s | 0.5317663 | 0.5319583 | 0.5354122 | 0.4354289 | 0.4345412 | 0.4378743 |
| Random Forest Regressor | 35.6357s | 0.5305861 | 0.5303797 | 0.5341357 | 0.4345269 | 0.4331568 | 0.4368884 |
| Gradient Boosting Regressor | 44.0408s | 0.5170131 | 0.5283126 | 0.5338617 | 0.4238456 | 0.4320921 | 0.4370584 |

1. Table 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE | Test Time |
| 3453.0686s | 0.444250 | 0.4780201 | 0.5367674 | 0.3575821 | 0.3788592 | 0.4274113 | 60.2s |

1. Table 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Train RMSE | Test RMSE | Train MAE | Test MAE |
| Random Forest – 100 estimators | 0.1985753 | 0.5212393 | 0.1471460 | 0.4221629 |
| Gradient Boosting Regressor scikit-learn | 0.4726019 | 0.5169296 | 0.3906506 | 0.4250207 |
| Linear Regression | 0.4578885 | 0.5269989 | 0.3706658 | 0.4253710 |
| XGBoost – after finetuning | 0.4309509 | 0.5144941 | 0.3569512 | 0.4214371 |

1. Table 5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Experiments\Metrics | Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE | Test Time |
| Old preproc, dropout=0.3 (128,64) lr=0.001 | 6475.2331s | 0.390383 | 0.484777 | 0.53174 | 0.3087084 | 0.388977 | 0.43025 | 277.6s |
| Old preproc, dropout=0.4 (128,64) lr=0.0001 | 6442.0399s | 0.440190 | 0.475838 | 0.53408 | 0.3525605 | 0.382628 | 0.43201 | 275.6s |
| Old preproc, dropout=0.5 (128,64) lr=0.001 | 5280s | 0.433440 | 0.476665 | 0.53017 | 0.3468251 | 0.384829 | 0.42982 | 289s |
| New preproc, wv (128,2), dropout=0.5 (128,64) | 1200s | 0.422961 | 0.474326 | 0.52799 | 0.3386136 | 0.378734 | 0.42482 | 60s |
| New preproc, wv (128,5), dropout=0.5 (128,64) | 1831.0647s | 0.451372 | 0.480993 | 0.52772 | 0.3642098 | 0.387826 | 0.42767 | 87.7s |
| New preproc, wv (128,5), dropout=0.5 (64,64) | 725.3534 | 0.472367 | 0.481953 | 0.52662 | 0.3825048 | 0.385214 | 0.42090 | 29.4s |
| New preproc, wv (128,5), dropout=0.5 (64,64) | 740.3159s | 0.487391 | 0.489541 | 0.52577 | 0.3955178 | 0.390812 | 0.42009 | 29.4s |
| New preproc, wv (128,6), dropout=0.5 (64,64) | 1750s | 0.481631 | 0.483944 | 0.52378 | 0.3904439 | 0.387263 | 0.41984 | 30.8s |
| New preproc, wv (128,7), dropout=0.5 (64,64) | 1564s | 0.476827 | 0.485717 | 0.52137 | 0.3883239 | 0.391543 | 0.41891 | 29.5s |

1. Table 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE |
| 2LSTM | 0.443442784 | 0.497304566 | 0.529487811 | 0.357731998 | 0.39828235 | 0.424925996 |

1. Table 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Train RMSE | Test RMSE | Train MAE | Test MAE |
| RF-100 | 0.2060765 | 0.5401440 | 0.1510049 | 0.4374540 |
| Gradient Boosting Regressor scikit-learn | 0.5111906 | 0.5295776 | 0.4202823 | 0.4343958 |
| Linear Regression | 0.4797027 | 0.5349621 | 0.3885695 | 0.4337574 |
| XGBOOST | 0.4227265 | 0.5222816 | 0.3497001 | 0.4265357 |

1. Table 8

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE | Test Time |
| Bert (256,64) | 36s | 0.3837024 | 0.4831213 | 0.5258223 | 0.3126069 | 0.38586 | 0.4220641 | 3.4s |
| bert (512,256,128) | 30s | 0.4010291 | 0.4794979 | 0.5233444 | 0.3285939 | 0.38809 | 0.4245823 | 3.2s |
| bert (512,256,128) adjusted lr | 33s | 0.3912655 | 0.4856261 | 0.5232600 | 0.3192868 | 0.39342 | 0.4243777 | 3.3s |
| bert (512,256,128) - best 0.519 | 30s | 0.4027725 | 0.4794790 | 0.5198253 | 0.3299059 | 0.38799 | 0.4223737 | 3.1s |

1. Table 9

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Training Time | Train RMSE | Val RMSE | Test RMSE | Train MAE | Val MAE | Test MAE | Testing Time | |
| Random Forest | 34.4749s | 0.530366 | 0.5302874 | 0.5340345 | 0.434309 | 0.433044 | 0.4367363 | | - |
| Gradient Boosting | 45.2196s | 0.516026 | 0.5281616 | 0.5338571 | 0.423372 | 0.431854 | 0.4371989 | | - |
| Character Based | 3453.069s | 0.439130 | 0.4812120 | 0.5367674 | 0.353062 | 0.379115 | 0.4274113 | | 60.2s |
| Word based | 1564s | 0.476827 | 0.4857175 | 0.5213677 | 0.388324 | 0.391543 | 0.4189073 | | 29.5s |
| 2LSTM | - | 0.443443 | 0.4973045657 | 0.5294878106 | 0.357732 | 0.398282 | 0.4249259965 | | - |
| XGBoost word pretrained | - | 0.430951 | - | 0.5144941 | 0.356951 | - | 0.4214371 | | - |
| Bert | 30s | 0.402773 | 0.4794790 | 0.5198253 | 0.329906 | 0.387999 | 0.4223737 | | 3.1s |
| XGBoost Bert pretrained | - | 0.422727 | - | 0.5222816 | 0.349700 | - | 0.4265357 | | - |

Figures

1. *Figure 1*

A diagram of a diagram

Description automatically generated

1. *Figure 2*

A diagram of a flowchart

Description automatically generated

1. *Figure 3*

A graph with lines and numbers

Description automatically generated

1. *Figure 4*

A graph with lines and numbers

Description automatically generated

1. *Figure 5*

A graph with lines and numbers

Description automatically generated

1. *Figure 6*

A graph with blue and orange lines

Description automatically generated

1. Figure 7

A graph with a line and a line

Description automatically generated

1. Figure 8

A graph with a line and a line

Description automatically generated