CS584: ESTIMATING SEARCH RELEVANCE USING MODERN DEEP NEURAL NETWORKS

Aditi Duggal[1]
[1]Stevens Institute of Technology
aduggal2@stevens.edu

ABSTRACT

Most of the websites today have search functionality to allow users to search for various documents such as products, user posts, text articles etc. It is important to suggest semantically relevant documents to user. The goal of this project to predict the relevance score given product information and search term. The relevance score describes how relevant a product is to the given search term. The idea is to use state of the neural network models to predict the relevance score. I plan to train various neural network models including Bidirectional LSTM, One-dimensional Convolution, Attention Mechanism and Transformer based models such as BERT to solve the problem. Then will evaluate each of the model using RMSE as a metric. It is also important to consider time taken to predict the relevance as a constraint while evaluating models.

1 Introduction

In the past few decades the e-commerce industry has been expanding tremendously. There have been provisions for users to buy whatever they need just by the click of a mouse or tap of the screen. Home Depot is one such online marketplace that allows its users to scan through and purchase the latest home improvement products ranging from home appliances to kitchen remodeling, bathroom decorating ideas to patio furniture and so on. However, in order to thrive amongst the ever-growing competition in the industry, providing a smooth user experience is of utmost priority. As a result, it is necessary to meet the customer expectations of obtaining accurate results to the queries typed in by them as quickly as possible.
This paper is based on an ongoing challenge on Kaggle posted by Home Depot. The goal of this project is to develop such a model that could accurately predict the relevance of search results, thereby helping them improve their customer's shopping experience. Home Depot makes use of the concept of search relevancy as a measure of how quickly their customers are able to find the products that they need. As of now, the search algorithms that are being used by Home Depot are analyzed by some designated human raters. However, evaluating the impact of different potential changes on the algorithm manually is a pretty daunting and time-consuming task. Through this project, Home Depot aims at decreasing the human input in the evaluation of the search relevancy, in order to make substantial increments in the number of iterations that can be performed by their team on the search algorithms being used at present.
In this paper, we have aimed at giving an overview of the different approaches that were adopted in order to derive a solution to the problem given in three main sections. In the first section, we discuss about the initial steps of our project that involved a thorough exploration of the datasets provided, followed by the pre-processing of the textual data. The next section deals with extracting various relevant features from the dataset and fit to different neural network models. Finally, in the last section we examine the results obtained by applying our approach and analyze how each of the methods trade-off against each other.

2 Background / Related work

Semantic similarity is one of the open research topics that have been discussed over the past decade. Evaluating the similarity between the text data is one of the arduous problems. The versatility of natural language makes it diffi-

cult to define rule-based methods for determining semantic similarity measures. The citation Evolution of Semantic Similarity[1] discusses semantic similarity methods introduced in past years.
Traditionally, Lexical similarity is used to find the relevant documents for the given search term. Keyword matching is used to find the relevant documents where the model look for literal matches of the query words or variants of them, without understanding the overall meaning of the query. To overcome this problem, semantic similarity is used to find the semantically relevant documents. It is a technique in which a search query aims to not only find keywords but to determine the intent and contextual meaning of the words a person is using for search.
The theory of semantic similarity goes as far back as 2003, and a paper written by R. Guha et al, Rob McCool, and Eric Miller, where they introduced Semantics Search[2] for Text data. Word Embedding was initially estimated using techniques such as latent semantic analysis to analyze relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. However, these techniques faced problems such as synonymy and polysemy. Advances in Deep Neural Networks were helpful in overcoming such problems by finding semantically valid word embeddings. Recently transformer-based models such as BERT, GPT2, XLNet to name a few have gained popularity as they outperform all existing model.

3 Experiment Design

3.1 Description of the dataset

The data has been provided by the hosts of this competition, Home Depot and the source is Kaggle[3] The data is spread across three main datasets: train, product descriptions and attributes. The train data consists of approximately 74k observations. The other two datasets are provided to us in order to give us detailed information about each of the Home Depot products.
The training set contains the products as well as the corresponding search terms that were used to retrieve the product results. Additionally, the training set also provides us with the relevance scores for each product-search pair. The relevance score describes how relevant a product is to a given search term. These scores are obtained by averaging the relevance ratings provided by three human raters manually and fall in a continuous range of 1 to 3. For example, The relevance is a number between 1 (not relevant) to 3 (highly relevant).For example, a search for "AA battery" would be considered highly relevant to a pack of size AA batteries (relevance = 3), mildly relevant to a cordless drill battery (relevance = 2), and not relevant to a snow shovel (relevance = 1) signifying highest relevance. For the purpose of validating the models, we decided to split the train set into 80:20 ratio, with 80 % data being used to fit the model and the remaining 20 % data being used as the test data. Our task is then to predict the relevance scores for the product-search pairs for the test data. Product description and attributes dataset contains useful information such as textual description and technical specifications of the products.

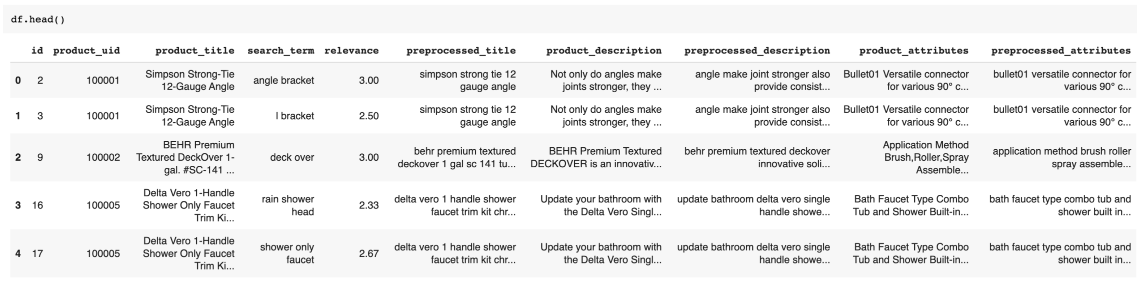
3.2 Prepossessing

The given information about each product is somewhat dirty and poor-structured. As a result, it's necessary to clean and convert the data into the desired format. For the prepossessing there was one main modification made to the training dataset in order to accommodate useful information from the product descriptions and attributes datasets.The product description and attributes were appended for each product based on its unique identifier,product id.
Once the dataset was modified as desired, I went ahead and performed some basic data preprocessing. Since the data that we were dealing with was mostly textual data, applying appropriate and adequate preprocessing techniques was found to be pretty crucial for this project. Further preprocessing was performed on the data in the following steps:

1. Since the dataset contained words with a mix of upper and lowercase letters, the first step was to format the dataset to contain only lowercase letters. This step helped us to standardize the dataset and perform further text processing easily.
2. Next, I went ahead and eliminate words that did not provide much useful information. So, all the punctuations and URLs.
3. I wanted to tokenize the words and removed the stop words using nltk.corpus. Finally, to reduce inflectional forms to a common base form I performed Lemmatization on words accordingly. For this nltk's WordNetLemmatizer class was used which is a thin wrapper around the wordnet corpus.

3.3 Feature Engineering

Feature engineering is important step when we deal with the text data which are relevant to each other. Some of the features were extracted from the dataset. Also, we can't pass textual data to a model. So we needed a way through



```
df.head()
```

	id	product_uid	product_title	search_term	relevance	preprocessed_title	product_description	preprocessed_description	product_attributes	preprocessed_attributes
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00	simpson strong tie 12 gauge angle	Not only do angles make joints stronger, they ...	angle make joint stronger also provide consist...	Bullet01 Versatile connector for various 90° c...	bullet01 versatile connector for various 90° c...
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50	simpson strong tie 12 gauge angle	Not only do angles make joints stronger, they ...	angle make joint stronger also provide consist...	Bullet01 Versatile connector for various 90° c...	bullet01 versatile connector for various 90° c...
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.00	behr premium textured deckover 1 gal sc 141 tu...	BEHR Premium Textured DECKOVER is an innovativ...	behr premium textured deckover innovative soli...	Application Method Brush,Roller,Spray Assemble...	application method brush roller spray assemble...
3	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	rain shower head	2.33	delta vero 1 handle shower faucet trim kit chr...	Update your bathroom with the Delta Vero Singl...	update bathroom delta vero single handle showe...	Bath Faucet Type Combo Tub and Shower Built-in...	bath faucet type combo tub and shower built in...
4	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	shower only faucet	2.67	delta vero 1 handle shower faucet trim kit chr...	Update your bathroom with the Delta Vero Singl...	update bathroom delta vero single handle showe...	Bath Faucet Type Combo Tub and Shower Built-in...	bath faucet type combo tub and shower built in...

Figure 1: Prepared Dataset

which we could convert text to numbers. Just doing this would also not suffice. We miss out on the interaction between search and product if we only pass their numerical data to any machine learning algorithm. We need features which could show how the product information and search query are related to each other. Below is described the overall feature engineering process before starting with modeling.

1. Text Encoding: Text encoding is a process to convert meaningful text into number/vector representation so as to preserve the context and relationship between words and sentences, such that a machine can understand the pattern associated with text and can make out the context of sentences.
   For this step, Keras tokenization method fit_to_texts() was used. The list of titles, attributes and description from the test data were the argument to extract features from the data.
2. Feature Extraction: Since the dataset was entirely textual, this was used to extract a lot of features from the data. Following is the list of features that were obtained:
   - Length of titles, search term and description: These features gave the number of words in the search query, product title and product description. This can be an important parameter in deciding the significance of the search query. If the length of the search query is long, it gives a lot of information about what kind of product is the user expecting. Hence the list of products that could be displayed as a search result to a longer query could be lot more significant than the results to a shorter query.
   - Vocabulary Length: The vocabulary length of the product titles, product description, product attributes and the search query will help us understand the corpus we are dealing with. It will help during modeling for the length of each corpus.
   - Following graph shows a plot of length of text vs number of data points.

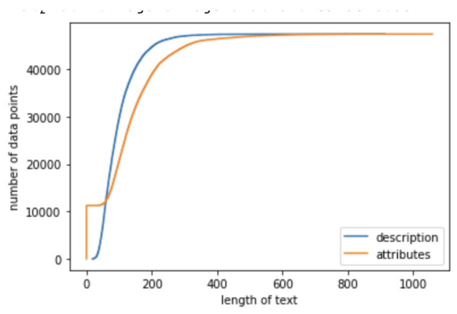


Figure 2: Length of text vs number of data points in dataset

3.4 Modeling and Learning

In this project, the task can be formulated as applying Deep Neural Network essentially and estimating the search relevance. After feature extraction, we need to fit the training data on a learning model and give predictions on the test data. Here I explore **b**three different models:

Bi-Directional LSTM

A Recurrent Neural Network is a network structure that can store historical states. However, due to vanishing gradient and gradient descent, multilayer RNN is limited when calculating context information. LSTM is a variation of the

RNN that mainly mitigates the problem of RNN long-distance gradient calculations. Mathematical representation for LSTM:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f$$
$$i_t = \sigma(W_{f^1} \cdot x_t + U_i \cdot h_{t-1} + b_i$$
$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o$$
$$c_t = f_t \circ c_t + i_t \circ tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c$$
$$h_t = o_t \circ tanh(c_t)$$

where x_t is current word embedding, W and U are the weights matrices, b is the bias h_t is hidden state. Further, i_t is input gate, f_t is the forget gate, c_t is the memory cell, σ is the sigmoid function, and $\cdot$ is the Hadamard product.
The improved Bi-LSTM-based can solve the problem of the inability of LSTM to calculate the context information of the reverse sequence. That is two LSTMs whose outputs are stacked together. One LSTM reads the sentence forward, and the other LSTM reads it backward. We concatenate the hidden states of each LSTM after they processed their respective final word. Two hidden states $h_t^{forward}$ and $h_t^{forward}$ from these LSTM units are concatenated into a final hidden state h_t^{bilstm}:

$$h_t^{bilstm} = h_t^{forward} \oplus h_t^{backward}$$

Where $\oplus$ is concatenation operator. The learning models based on LSTM for semantic relationship classification and found that BiLSTM can discover richer semantic information and make full use of contextual information than LSTM. BiLSTM obtains high-level semantic information features from word embedding and completes sentence-level relationship classification.
Following is the architecture of proposed model:

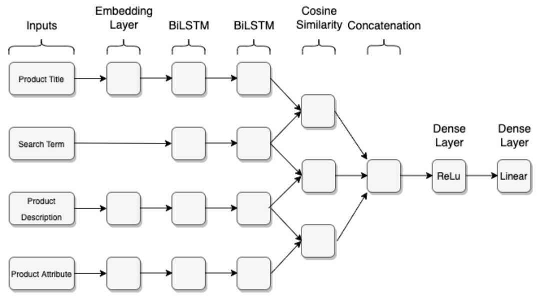


Figure 3: Bi-LSTM Architecture

Attention Mechanism

Attention mechanism is one of the greatest break through in the field of NLP. It is a powerful mechanism developed to enhance the performance of the neural network-based question answering, machine translations, speech recognition, to image captioning tasks.
A model based on the attention mechanism can quickly scan the global information and obtain a target area. Then, the attention resources are devoted to obtaining detailed information about the target and suppressing other useless information. By inputting the decoding vector into the attention layer, the attention mechanism can focus on the local part of the decoding vector.

$$M = tanh(H)$$
$$\sigma = softmax(w^T \cdot M)$$
$$r = H \cdot \sigma^T$$

In this section, we propose the attention mechanism. Let H be a matrix consisting of output vectors$[h_1, h_2, ..., h_T]$ that the LSTM layer produced, where T is the sentence length. The representation r of the sentence is formed by a weighted sum of these output vectors.

Above is the architecture of proposed model and the reference of BahdanauAttention is used to implement the attention layer.

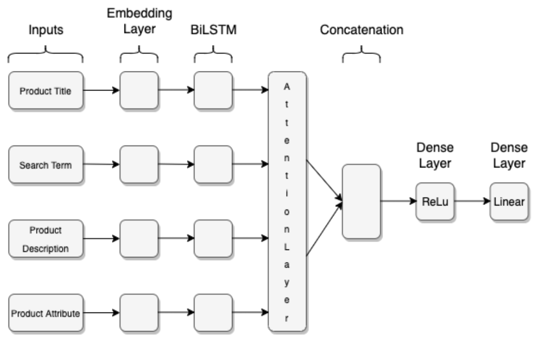


Figure 4: Attention Architecture

Conv1D Model

The application of convolutional neural networks is the same as in image data. The only difference is that 1D convolutions are applied instead of 2D convolutions. In images, the kernel slides in 2D but in sequence data like text data the kernel slides in one dimension. In the simplest case, the output value of the layer with input size $(N, C_{in}, L) and output(N, C_{out}, L_{out})$ can be precisely described as:

$$out(N_i, C_{outj}) = bias(C_{outj}) + \sum_{k=0}^{C_{in}-1} weight(C_{outj}, k) \circ input(N_i, k)$$

where, $\circ$ is valid cross-correlation operator, N is a batch size, C denotes a number of channels, L is a length of signal sequence.
The 1D convolution network will be made of the following: An embedding layer that turns the data into dense vectors of fixed size. More on this later.

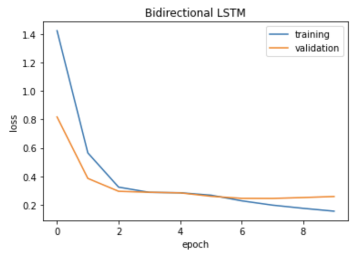
- A Conv1D with relu activation function.
- A GlobalMaxPooling1D layer that down samples the input by taking the maximum value.
- A Dense layer with the fully connected layer.
- An output layer with the activation function

BERT

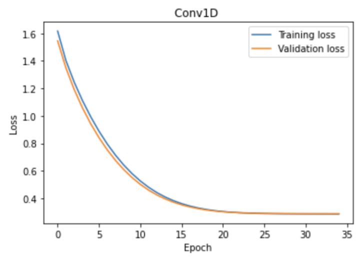
BERT is a pre-trained transformer-based word embeddings which can be fine-tuned by adding a final output layer to accommodate the embeddings to different NLP tasks. BERT uses the transformer architecture proposed by Vaswani et al, which produces attention-based , Vol. 1, No. 1, Article . Publication date: February 2020. 10 D Chandrasekaran and V Mago word vectors using a bi-directional transformer encoder. The BERT framework involves two important processes namely 'pre-training' and 'fine-tuning'. The model is pretrained using a corpus of nearly 3,300M words from both the Book corpus and English Wikipedia. Since the model is bidirectional in order to avoid the possibility of the model knowing the token itself when training from both directions the pre-training process is carried out in two different ways. In the first task, random words in the corpus are masked and the model is trained to predict these words. In the second task, the model is presented with sentence pairs from the corpus, in which 50 percent of the sentences are actually consecutive while the remaining are random pairs. The model is trained to predict if the given sentence pair are consecutive or not. In the 'fine-tuning' process, the model is trained for the specific down-stream NLP task at hand. The model is structured to take as input both single sentences and multiple sentences to accommodate a variety of NLP tasks.

4 Experimental results

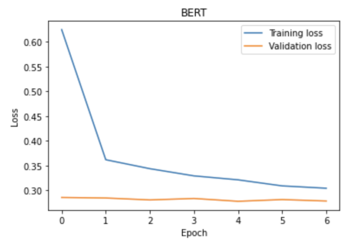
We trained every model and hyper-tuned the models. The training was done on 80% of the dataset. We then tested every model on the remaining 20%, which formed our testing data. Below shows the training and validation loss graphs on each epoch. This is the table for Evaluation metrics:



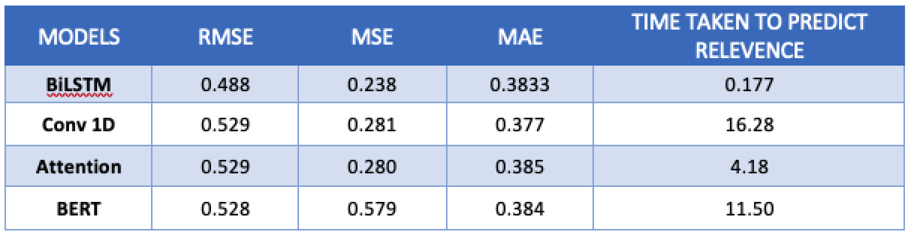
(a) Bi-LSTM



(b) Conv1D



(c) BERT



MODELS	RMSE	MSE	MAE	TIME TAKEN TO PREDICT RELEVENCE
BiLSTM	0.488	0.238	0.3833	0.177
Conv 1D	0.529	0.281	0.377	16.28
Attention	0.529	0.280	0.385	4.18
BERT	0.528	0.579	0.384	11.50

Figure 6: Evaluation metrics

5 Conclusion and Future work

1. I have trained various neural network models including Bidirectional LSTM, One-dimensional Convolution, Attention Mechanism, and Transformer based models such as BERT to solve the problem
2. Until now we got some good RMSE score of 0.488 from Bidirectional LSTM which has taken the least time of 0.177
3. I believe with better GPU, we can achieve a Sentence-BERT which would give better results.
4. With better feature extraction, we could achieve better RMSE score.
5. We can display the top 10 or 20 products with higher relevancy score to the user. while building such search engines is not the scope of this project, we can use trained models to build such systems. Building such systems will help us display semantically similar products to the user.

6 References

1. DHIVYA CHANDRASEKARAN and VIJAY MAGO, Evolution of Semantic Similarity arXiv:2004.13820v2 [cs.CL] 30 Jan 2021
2. Data Link: https://www.kaggle.com/competitions/home-depot-product-search-relevance/data
3. P. Zhou et al. "Attention-based bidirectional long short-term memory networks for relation classification," in Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers) 2016, pp. 207–212.
4. D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," arXiv Prepr. arXiv1512.01100 2015.
5. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In NIPS

References