UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Giovanni Paolo Santos de Carvalho

Using structured and unstructured data for product price prediction

Recife

2020

Giovanni Paolo Santos de Carvalho

Using structured and unstructured data for product price prediction

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador: Luciano de Andrade Barbosa

Coorientador: Tsang Ing-Ren

Recife

2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

C331u Carvalho, Giovanni Paolo Santos de
Using structured and unstructured data for product price prediction/ Giovanni Paolo Santos de Carvalho. – 2020.
64 f.: il., fig., tab.

Orientador: Luciano de Andrade Barbosa.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências.

1. Inteligência computacional. 2. Otimização. I. Barbosa, Luciano de Andrade (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2021 - 14

Giovanni Paolo Santos de Carvalho

“Using structured and unstructured data for product price prediction”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 17 de janeiro de 2020.

I dedicate this work to my girlfriend, Priscila, who has gifted me with her support and trust during stressful times, helping me to focus on my objectives and overcome the obstacles.

ACKNOWLEDGEMENTS

I thank my friends and family for their love and support. I am also grateful for my advisor, Luciano de Andrade Barbosa, and my co-advisor, Tsang Ing-Ren, for their knowledge, guidance, and encouragement.

ABSTRACT

Product price estimation is a relatively new trend in e-commerce that helps customers in their decision making process of buying or selling a product, giving a starting point of what could be a fair price. In this work, we are particularly interested in performing price prediction from online product offers. These offers usually present some text describing the product in natural language (unstructured data) and the specification of the product composed of its properties (structured data). In this dissertation, we aim to predict the price of product offers based on both structured and unstructured information. For that, we propose an attention-based network that deals with structured data individually, and also the interaction between this data and unstructured data, combining them to perform the prediction. For the structured information, we apply a regular fully-connected network; and to model the interaction between them (product's properties and its description), we employ a co-attention network. Those networks are combined and used by a neural network regressor to learn a vector representation of the product offer. This vector can then be used as a feature set by any regressor to perform product price prediction. This architecture is designed to operate with general structured and unstructured types of product offers, and in this particular study, it is evaluated on a car price prediction task, for which we collected a dataset by scraping 11 sources of car classifieds. Our experimental evaluation shows that: (1) regressors using the learned embedding obtained the best results, improving their performance in almost all scenarios in comparison to raw features; and (2) simple linear regressor models such as Linear Regression using the learned embedding achieved comparable results to more competitive algorithms such as LightGBM.

Keywords: Structured data. Unstructured data. Attention networks. Price prediction.

RESUMO

Predição automática do preço de produtos é uma tendência relativamente recente que ajuda indivíduos no seu processo de decisão a respeito de realizar uma compra ou uma venda de um produto, fornecendo um ponto de referência de qual seria um preço justo. Neste trabalho, estamos particularmente interessados em realizar a predição a partir de anúncios de produtos disponíveis na web. Esses anúncios frequentemente são acompanhados de algum texto descrevendo o produto em linguagem natural (dados não-estruturados) e de especificações do produto contendo as suas propriedades (dados estruturados). Nesta dissertação, visamos predizer o preço de um produto anunciado a partir de ambas modalidades de dados disponíveis. Para este fim, propomos uma rede baseada em atenção que lida com dados estruturados e também a interação entre esses e dados não-estruturados, combinando-os para realizar a predição do preço. Para os dados estruturados, utilizamos uma rede *Multilayer Perceptron* simples; e para modelar a interação entre ambos (descrição do produto e suas especificações), nós utilizamos uma rede com um mecanismo de *co-attention*. Essas redes combinadas são utilizadas em um regressor baseado em Redes Neurais para aprender representações vetoriais (*embeddings*) do produto anunciado. Este *embedding* pode ser utilizado como conjunto de características por qualquer regressor para realizar a estimação do preço. Esta arquitetura é projetada para operar com dados genéricos estruturados e não-estruturados de anúncios de produtos e, neste estudo em particular, ela é avaliada na tarefa de predição do preço de anúncios de automóveis na web, para a qual realizamos a coleta a partir de 11 sites de anúncios classificados. Nossos resultados experimentais mostram que: (1) regressores utilizando os *embeddings* aprendidos pela rede proposta obtiveram os melhores resultados, melhorando sua performance em quase todos os cenários em comparação com o conjunto original de dados; e (2) modelos de regressão mais simples como *Linear Regression* utilizando as características aprendidas alcançam resultados comparáveis a outros algoritmos mais competitivos como *LightGBM*.

Palavras-chaves: Dados estruturados. Dados não estruturados. *Attention networks*. Predição de preço.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 PROBLEM AND MOTIVATION

Exchanging currency for goods has been an integral part of our day to day lives for centuries. This exchange or trade is a fundamental concept that is required for supplying our most basic needs, such as food and shelter, up to other demands related to the quality of life or even luxurious wishes, such as home appliances, electronics, or toys.

Worldwide, e-commerce sales represented 2.98 trillion dollars[1] in 2018, and is expected to surpass the 5 trillion mark by 2022. When taking total retail sales into account, this figure is much larger. As with many other things, the popularization of the Internet-enabled users to not only consume more content but produce it as well. With online stores, the buyer can compare many more options than it would be feasible to do by visiting multiple locations. When selling, online marketplaces open up a much wider range of possible buyers to the seller. With a much larger scale, however, comparing products and determining its price becomes much harder. The seller does not want to devalue their product to be able to compete with other parties but also wants to maintain a good margin of profit. The buyer might wish to obtain the best deal available, but there might be many alternatives to the same type of product. Even for the same brand and model, it can still be difficult to evaluate whether the difference in features justifies the difference in prices. Knowing all the intricacies about what defines the price of even a small subcategory of products requires some amount of expertise.

The characteristics of a product such as its brand, which features it has available, the product's condition (new or used, and how much usage it has endured) are important in determining its price. Additional aspects, such as sentimental value and whether the product is rare or limited, such as collector's editions, also play a significant role.

Due to these mentioned factors, correctly pricing products is a hard challenge for humans. A growing trend is to use models to estimate the price of a product automatically. In this direction, e-commerce websites have started to offer price prediction tools to their users (buyers or sellers) to help them in this decision making process. This feature is particularly useful for sellers to offer price suggestions and for buyers to compare the price of a selling product to its estimated value. Kelley Blue Book[2] website, for instance, provides such a feature with its

[1] https://www.emarketer.com/content/global-ecommerce-2019
[2] www.kbb.com

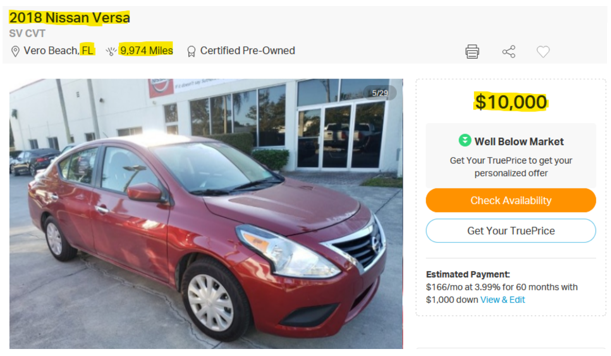
Price Advisor tool that estimates the market value of cars. The traveling website Kayak[3], also offers a kind of price prediction by suggesting to the users to book a flight or wait based on its forecast price model. Recently, the e-commerce website Mercari[4] launched a competition on Kaggle[5] aiming to solve the automatic product price estimation problem of its products.

In the context of online listings of products, there are usually some fields which are related to the product's attributes, e.g. model, color, storage capacity, screen resolution, camera resolution, etc for smartphones, and some fields dedicated to the listing title and its description in the form of text. Depending on the domain and where the listing is posted, those textual fields might contain additional information that is relevant to the asked price but is not present in the attributes which are more easily skimmed through and compared. With a wide range of available options, manually doing this comparison is arduous and time-consuming. In the context of used cars, for example, the description might contain details about whether the engine has been replaced or if any other sort of repair has been performed, which might have a direct impact on the depreciation of the vehicle. In other cases, in contrast, the description might be mostly advertisement or other info about the car dealership that posted the listing, or not contain other additional information that is not present in the structured attributes. As an example, in Figure 1 we show a listing from one of the car dealer site[6]. The attributes of the car such as year, manufacturer and US state where it is being sold are shown in Figure 1(a) and its description in Figure 1(b). Note that there is complementary information in the description that is not present in a structured form such as information about its breaks (ABS breaks) and seats (adjustable front bucket seats), and some of the structured information is also present in the description text, such as the manufacturer, model and year. We observe that while naively adding textual features might improve the performance of our models, targeting the information overlap and exploring co-occurrences between the two modalities leads to an even better improvement.

As previously exemplified, there can be relevant information present in the textual information (e.g. title and description) of a product that can be leveraged for product price estimation. Hand-engineered features or heuristics that indicate the presence or absence of specific characteristics or words might be too time-consuming, and too domain-dependent: features of smartphones do not apply to cars, for example. A simple approach when dealing with the

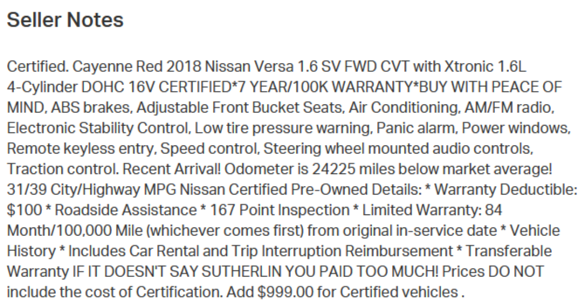
[3] www.kayak.com
[4] https://www.mercari.com/
[5] https://www.kaggle.com/c/mercari-price-suggestion-challenge
[6] As extracted from the URL archived at <https://archive.is/p4fiQ>

Figure 1 – Example of a car listing from the TrueCar website. In the left, the type of information that was extracted from the page during the scraping process was highlighted. In the right, the entire text in the "Seller Notes" field was extracted.
Source: Authors' own elaboration.


2018 Nissan Versa
SV CVT
Vero Beach, FL 9,974 Miles Certified Pre-Owned
$10,000
Well Below Market
Get Your TruePrice to get your personalized offer

Estimated Payment: $166/mo at 3.99% for 60 months with $1,000 down View & Edit

(a) Listing attributes.


Seller Notes

Certified. Cayenne Red 2018 Nissan Versa 1.6 SV FWD CVT with Xtronic 1.6L 4-Cylinder DOHC 16V CERTIFIED*7 YEAR/100K WARRANTY*BUY WITH PEACE OF MIND, ABS brakes, Adjustable Front Bucket Seats, Air Conditioning, AM/FM radio, Electronic Stability Control, Low tire pressure warning, Panic alarm, Power windows, Remote keyless entry, Speed control, Steering wheel mounted audio controls, Traction control. Recent Arrival! Odometer is 24225 miles below market average! 31/39 City/Highway MPG Nissan Certified Pre-Owned Details: * Warranty Deductible: $100 * Roadside Assistance * 167 Point Inspection * Limited Warranty: 84 Month/100,000 Mile (whichever comes first) from original in-service date * Vehicle History * Includes Car Rental and Trip Interruption Reimbursement * Transferable Warranty IF IT DOESN'T SAY SUTHERLIN YOU PAID TOO MUCH! Prices DO NOT include the cost of Certification. Add $999.00 for Certified vehicles .

(b) Listing description.

price prediction task would be to only use the structured attributes and classical learning algorithms such as linear models, penalized linear models (Ridge (HOERL; KENNARD, 1970), Lasso (TIBSHIRANI, 2016), ElasticNet (ZOU; HASTIE, 2005)) or ensemble methods (Random Forests (BREIMAN, 2001), LightGBM (KE et al., 2017)).

In this work, we aim to estimate the asking price of products - vehicles, specifically - based solely on the information available in the online product listings (structured and unstructured). It is important to mention that we perform product price prediction for a specific snapshot of products in time rather than a continuous price estimation task (forecasting). The two modalities, structured attributes and unstructured text, present in the product listing have differences. The former is usually cleaner but might miss out on some details that were not specified as a field in the listing. The latter might contain such details but usually has more noise. The main challenge is how we leverage the information contained in both structured and unstructured variables and use one to add additional context to the other, in such a manner that the overall performance is greater than a traditional combination of both.

To this end, we propose a deep learning network that learns patterns from those two modalities to perform product price prediction. The model is composed of two different branches. The first branch applies an MLP network to deal with the structured data. The second one uses a co-attention network to model the complementarity and interaction between the structured and unstructured information. Essentially, the co-attention network learns how interactions between the two modalities can be leveraged to improve the predictive power of each by attending to more relevant parts of the text according to the structured attributes of the products and vice-versa, and also learning from the text complementary information for the product price estimation. That is, instead of extracting those features directly, either manually or through heuristics, they are learned jointly with the network for the task. The representations generated by the structured and co-attention branches are then fed into an MLP that applies a regressor layer to perform product price prediction.

The textual inputs in our network, rather than being treated as histograms of word occurrences, such as in the bag-of-words model, is encoded into a denser and order-of-magnitudes smaller vectorial representation (embeddings). The parameters of these encodings are part of the network parameters that are updated during the training process. Structured categorical variables go through the same treatment, and for the numerical structured attributes, they are normalized according to parameters which are also part of the trainable weights in the network.

Looking at the complete architecture, it is an end-to-end encoder of structured and unstructured features that generates a representation of product offers that is mapped to some output by a task-dependent function. In the context of this work, we output a single real-valued prediction which is the price of the product according to its features. However, the price prediction task can be thought of as an auxiliary tool to generate this representation by the optimization of the network's parameters, that is used to define an objective function for the training process. Other mechanisms could also be used for performing representation learning: such as reconstructing the original data in an encoder-decoder fashion (CHANDAR et al., 2016) or in a classification task. The purpose of the network is merely defined by the behavior of the last layer and the loss function that is employed during the training. In our case, the final (output) layer encodes the outermost feature map into a price prediction. This representation, therefore, encodes the relevant aspects of all of the intermediate mappings and non-linearities that were performed in the previous layers of the network. The complete architecture, short of the last layer, is similar to a feature extractor that applies a transformation on the original raw data, and this representation can then be used to train other algorithms on the same regression task or in other related ones.

We carried out experimental evaluations on a real-world dataset that was collected from multiple sources of car listing websites based in the US. Our results show that:

- The proposed network outperforms competitive models such as Random Forest and LightGBM using the same features;

- The learned embeddings further improve the performances of the evaluated regressors in comparison to the same regressors trained using the raw features, meaning the network is able to encode additional information to facilitate the problem;
- The distinct evaluated regressors have comparable performance using the learned embeddings generated by the proposed model, meaning there are fewer trade-offs when choosing one or the other since the features are already of good quality.

1.2 OBJECTIVES

The main objective of this work is to effectively use structured and unstructured information of product offers such that the model can learn useful representations that perform better in the price prediction task compared to raw features. We perform an evaluation of these learned representations in the domain of online vehicle listings.

To attain this goal, the following tasks are required:

- Find and scrape car classified sites to extract structured and unstructured attributes that are common among the sources, and in addition, review, clean and process the data as to deal with extraction abnormalities, outliers, information leakage, etc;
- Build a deep neural network geared with architectural details that enable the modeling of interactions between structured and unstructured variables, such that the final performance is improved in comparison to naively using a combination of both;
- Generate from the raw features a dense representation that encodes not only the original information, but additional enriched interactions into a lower-dimensional space
- Compare the performance of traditional regression algorithms with the original feature space and our learned representation.

1.3 WORK ORGANIZATION

In this section, we outline the organization for the remainder of this work. In Chapter 2, we introduce and discuss the concepts and terminology which are the basis for the development of this work. We overview natural language-specific concepts, such as how to handle non-numerical data. In the following section, we discuss the relevant topics of artificial neural networks with a brief contextualization of the history of attention mechanisms. Lastly, we cover the details of representation learning and its various forms and applications.

In Chapter 3, we discuss studies that are relevant to the one conducted in this work. more specifically, we consider three broader groups in which they fall: a) text regression models, b) representation learning models and c) co-attention-based models.

In Chapter 4, the product of this research, the attention-based neural network, is introduced and explained in detail.

In Chapter 5, we discuss the data that was collected and which steps were taken to properly clean it. We go into feature sets which are considered in our experimental setup, as well as other baseline classical regressors. Furthermore, we also discuss hyperparameter settings and technical implementation details, as well as evaluation metrics. Lastly, we present the results and a discussion of what was observed from the experiments.

In Chapter 6, we conclude this work with an overview of its contributions and future research directions.

2 CONCEPTS AND TERMINOLOGY

In this section, we introduce the concepts this work builds upon. Since the proposed solution revolves around an attention mechanism, we focus on the building blocks that are necessary to grasp how these mechanisms contribute to the overall performance.

2.1 REGRESSION

In Goodfellow, Bengio and Courville (2015), a machine learning algorithm is defined as an algorithm that is capable of improving the performance of a computer program at some specific task through supervised or unsupervised experience. In this study, our focus is on the supervised learning experience. This computer program constitutes a machine learning model, and the learning process is what optimizes its performance at the task. The task of product price prediction is a regression task that can be seen as a function $f : \mathbb{R}^n \to \mathbb{R}$ of some inputs that outputs a real-valued number. Instead of defining the parameters of the function, we define the function and the optimization algorithm figures out what parameters of this function minimize the cost function (more on this in Section 2.2.1).

2.1.1 Linear Regression

The most basic model for the regression task is the Linear Regression model. The linear model is a function that outputs a linear transformation of its inputs

$$\hat{y} = \sum_{i=1}^{n} w_i x_i \, , \tag{2.1}$$

where w_i is the coefficient associated with the input feature x_i, and $\hat{y}$ is the output of the function. These coefficients are the parameters of the model, and they control the importance of each feature to the final prediction. Positive coefficients mean that a particular feature has a positive contribution to the output, while negative coefficients mean that a particular feature is negatively correlated with the output. This computation is often expressed in terms of the dot product between a vector $\mathbf{w}$ of parameters and an input vector $\mathbf{x}$

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} \tag{2.2}$$

To find the best set of parameters, a performance measure is needed. A common way of measuring the performance of regression models is by computing the mean squared error between the actual values and the values as predicted by the model. Considering a vector y of ground truth values, a vector $\hat{y}$ of predictions and a vector of errors (residuals) $e = \hat{y} - y$, all of size m, the MSE is expressed by

$$\text{MSE} = \frac{1}{m} \sum e_i^2 \tag{2.3}$$

This expression is equivalent to the square of the L2-norm of the residual vector, divided by the number of observations m

$$\text{MSE} = \frac{1}{m} \|e\|_2^2 \tag{2.4}$$

Naturally, minimizing the L2-norm of the vector of residuals, i.e. the Euclidean distance between the vector of predictions and the vector of targets, results in a model with lower MSE error. The set of parameters that minimizes this cost function can be found in closed form by solving for where its gradient is 0. An extension to this learning algorithm includes an intercept term that is added to the resulting multiplication of each feature and coefficient pair. This allows the algorithm to model for each feature x_i a line in the form of $w_i x_i + b_i$ such that this line does not necessarily have to go through the origin. This intercept is known as the bias and does not need to be fixed. It is learned automatically by adding one extra feature set to the vector of features x instead of adding a separate parameter. This way, the weight associated with this extra feature serves the purpose of the bias term.

2.2 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks extend Linear Regression models in terms of two characteristics: non-linearities and chaining of multiple functions, although there are non-linear regression models that are not neural networks.

The most basic unit of a NN, also known as a Neuron, is a composition of a linear mapping and a non-linear activation function

$$f(x) = \phi(w \cdot x + b)\,, \tag{2.5}$$

where w and b are the learnable parameters, the weights and the biases, and ϕ is a function that confers the power of learning non-linear relationships between the neuron's input features. Examples of such functions are sigmoid, hyperbolic tangent and rectified linear units (GLOROT; BORDES; BENGIO, 2011).

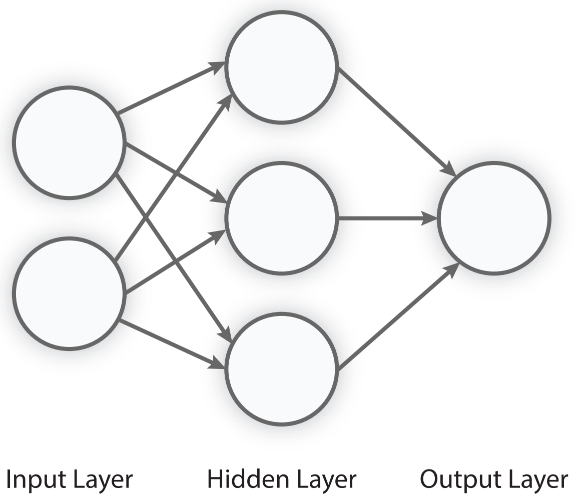
2.2.1 Optimization

With the usage of non-linear activation functions, most loss functions will result in a non-convex error space (GOODFELLOW; BENGIO; COURVILLE, 2015). As such, linear equation solvers are not suitable and gradient-based iterative algorithms are more commonly used. However, given the number of parameters of most neural networks and the number of training samples in most Deep Learning applications, the memory requirements for using linear solvers would be too high. The back-propagation (RUMELHART; HINTON; WILLIAMS, 1986) is commonly used to compute the gradient of the cost function w.r.t the network's parameters. With the computed gradient, the parameters are updated in the opposite direction, scaled by some learning rate. Each parameter update is called a step. Instead of simply subtracting the gradient times the learning rate from the parameters, more sophisticated strategies add a fraction of the previous step direction, such as momentum (SUTSKEVER et al., 2013) or by keeping a separate adaptive momentum and learning rate for each parameter, such as Adam (KINGMA; BA, 2015).

2.2.2 Multilayer Perceptron

In Multilayer Perceptron (MLP) networks, multiple neurons are stacked in a layer, effectively producing multiple non-linear mappings of the original input vector. The architecture of MLPs is composed of multiple layers. The non-linear mapping of the multiple neurons in the first layer, the input layer, generates an intermediate representation of the original feature space. The number of neurons in the layer determines the dimensionality of this representation. The next layers, the hidden layers, work with additional levels of intermediate representations. The last layer, the output layer, maps the representation into the predicted output. In the case of regression, the last layer contains a single neuron. In Figure 2, we present a simple MLP network with two neurons in the input layer, 3 neurons on the hidden layer and the output layer with a single neuron.

Figure 2 – Architecture of a Multilayer Perceptron network with two input features, a single hidden layer with two neurons and an output layer with one neuron.
Source: Authors' own elaboration.



2.2.3 Recurrent Neural Networks

A regular MLP can be used to model problems involving sequences, such as predicting the word that succedes a sequence of words (language modeling). However, the problem formulation is adjusted, for example by limiting the input to a fixed-size window context of the previous words. Recurrent Networks (RNNs) (RUMELHART; HINTON; WILLIAMS, 1987) tackle sequential data without such adjustments. Consider a sequence $\mathbf{x} = x_i, ..., x_n$ as a single sample composed of n elements, where n might be different for different samples. An RNN computes a state for each element x_i in the sequence as a function of the previous state

$$h_t = f(h_{t-1}, x_t) \tag{2.6}$$

The state for the first element in the sequence might be initialized with zeroes or random values. Here, f is a function defined by a neural network, such as an MLP. When training, the state computation is insufficient to generate the output, but combining it with an output layer that matches the characteristics of the training task allows for the last state h_n to serve as serve as a representation for the sequence. Since different training samples have different length, the recurrent part of the whole network is unfolded to match the length of the input. This dynamically-sized network can be seen as a network where layers have shared parameters. In practice, computations over large sequences are impractical both from the computational standpoint as well as the optimization process, because gradients become very small as the error is propagated back through the network (vanishing gradient) (HOCHREITER, 1991). As an attempt to mitigate these problems, the Long Short-term Memory network (HOCHREITER; SCHMIDHUBER, 1997) learns more advanced functions to control its hidden state. The next state computation is give by

$$\begin{aligned}
f_t &= \sigma_g \left(W_f x_t + U_f h_{t-1} + b_f\right), \\
i_t &= \sigma_g \left(W_i x_t + U_i h_{t-1} + b_i\right), \\
o_t &= \sigma_g \left(W_o x_t + U_o h_{t-1} + b_o\right), \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c \left(W_c x_t + U_c h_{t-1} + b_c\right), \\
h_t &= o_t \circ \sigma_h \left(c_t\right)
\end{aligned} \tag{2.7}$$

where f, i and o are gating functions that control how much to forget, input into and output from the hidden state h_{t-1} and current element x_t to the internal state c_t and output state h_t. There are also U and W weight matrices and a bias b associated with the computation of each gating function. Finally, $\sigma_g(z)$ is a sigmoid activation function that scales the activation z to 0,1, σ_c and $\sigma_h(z)$ are hyperbolic tangent (tanh) activation functions that scale the activation z to -1,1, and $A \circ B$ is the Hadamart product (element-wise) between A and B.

There are other types of RNNs with similar mechanisms, such as the Gated Recurrent Unit (CHO et al., 2014).

2.2.4 Attention Mechanisms

When dealing with variable-length input data, Recurrent Neural Networks (RNN) are a natural way of modeling sequences. Attention mechanisms were initially designed to alleviate a limitation of encoding representation in sequence to sequence models (BAHDANAU; CHO; BENGIO, 2015).

Most of the early neural translation models used the encoder-decoder framework (CHO et al., 2014; SUTSKEVER; VINYALS; LE, 2014), where it is first used an RNN to encode the sequence and generate a representation, and then a decoder RNN to generate the output sequence in the target language.

Consider the sequence of vectors $\mathbf{x} = (x_i, ..., x_m)$ where x_i corresponds to the i-th word in a sequence of size m, and the target sequence of words $\mathbf{y} = (y_i, ..., y_n)$, where y_i is the i-th word in a sequence of size n. The translation task is defined as a function $\mathbf{y} = f(\mathbf{x})$.

The encoder-decoder framework uses an encoder f that generates the encoding for each word in the input sequence $h_t = f(x_t, h_{t-1})$, then reduces this sequence of states to a context vector $c = q(h_1, ..., h_m)$. Then the decoder g also decodes sequentially, generating the predictions for the works in the target sequence $y_t = g(y_{t-1}, s_t, c)$, with current state s_t, context c and previous word y_{t-1}.

Instead of using the last hidden state of the encoder network h_m to generate the context vector c as in Cho et al. (2014), the hidden states at all steps along the sequence can be combined, as proposed in Bahdanau, Cho and Bengio (2015). More specifically, at each step, the decoder network is fed with the previous state and a dynamic context vector c_i generated by a feed-forward network on every state generated by the encoder, such that c_i is different at each step

$$c_i = \sum_{j=1}^{t} \text{Softmax}(a(s_{i-1}, h_j)) \cdot h_j \,, \tag{2.8}$$

where h_j is the concatenation of hidden states of a bidirectional encoder, and a, an MLP, computes an alignment score between current input j and output i, that is normalized to a probability using the Softmax activation function.

This work spawned a series of generalized attention mechanisms with different methods for computing attention scores, such as the general attention (LUONG; PHAM; MANNING, 2015), where the attention scores are computed as $s_t^\top W\mathbf{h}$, where s_t is the current decoder hidden state, $\mathbf{h}$ is the matrix of hidden states from the encoder and W is a matrix with learnable parameters. For modeling relationships between different tokens in the same sequence, the attention scores can be computed by replacing the target sequence with the input sequence. This is known as self-attention (CHENG; DONG; LAPATA, 2016).

Further improvements in the sequence to sequnce task were achieved with the Transformer model (VASWANI et al., 2017), which does not rely on recurrent mechanisms, but rather on pure self-attention. In the Transformer model, the multi-head attention is a general attention mechanism that computes the attended output based on three input vectors $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ associated with a query, a key and a value. It is computed by calculating the scaled dot product attention scores between the $\mathbf{q}$ and $\mathbf{v}$, normalizing with softmax to generate a probability distribution and finally multiplying the attention scores with $\mathbf{v}$:

$$\text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathrm{softmax}\left(\frac{\mathbf{q} \cdot \mathbf{k}}{\sqrt{n}}\right)\mathbf{v} \tag{2.9}$$

Additionally, this computation is done in projections of the original space, computed by learnable parameter matrices which transform the input into a subspace, allowing the model to learn more advanced relations. Furthermore, these heads are independent and can be computed in parallel.

$$\begin{aligned}\text{MultiHead}(\mathbf{q},\mathbf{k},\mathbf{v}) &= [\text{head}_1;\ldots;\text{head }_h]\,\mathbf{W}^o\,,\\ \text{head}_i &= \text{Attention}\left(\mathbf{q}W_i^q,\mathbf{k}WW_i^k,\mathbf{v}W_i^v\right)\end{aligned} \tag{2.10}$$

Where W_i^q, W_i^k and W_i^v are the learnable projections specific to the i-th head, and W^o a projection that combines the output from the parallel attention heads. The projection matrices are defined as to map the original subspace to d/h in a model where the input vectors have d-sized inputs and a total of h heads. Naturally, using the same input sequence as the query, key and values constitutes self-attention, although this generalization allows for modeling attention between different sequences.

In Lu et al. (2016) the authors introduce the concept of co-attention. They propose a mechanism that jointly attends to text and visual modalities for the Visual Question Answering (VQA) task. While their main contribution is on the hierarchical processing of the question in three levels (word, phrase and entire question), their results show that the co-attention allows the proposed network to attend to different regions in the image as well as different fragments of the question. This work builds upon Santos et al. (2016), extending their attention mechanism, used for question answering (not visual), which at the time was not introduced as co-attention, with two types of co-attention calculation parallel (on both image and question simultaneously) and alternating between generating the attentions for the image and the question. Another derived work is Xiong, Zhong and Socher (2019) which employs the co-attention encoder for the Question Answering task and introduced a dynamic decoder responsible for estimating the answer span.

Another model which does not introduce itself as a co-attention mechanism, but performs a similar computation is described in Yin et al. (2016), where three attention schemes based on CNN encoders of bigrams (word-pairs) are proposed. A more recent work, Hu et al. (2018), employs a co-attention mechanism for attending meta-paths in a network (Heterogeneous Information Network) associating users and movies and for a movie recommendation system. More specifically, the co-attention mechanism mutually enhances the embeddings for users, movies, and meta-paths. Some works which also have similar mutual-enhancing mechanisms for representations are used in Computer Vision for semantic segmentation (LIU; YIN, 2019;

HUANG et al., 2018) under the name of cross attention. In general, co-attention is not a widespread definition in the families of attention mechanisms. Moreover, it has been used under other names. For the remainder of this work, we consider a co-attention mechanism to be a type of attention that uses representations (embeddings) from two or more different modalities, e.g. image and text in VQA, document text and question text for QA, to produce enriched co-dependent representations for each. In this work, our co-attention module enhances the representation of structured categorical variables (non-sequential) and unstructured text (sequential).

2.3 LEARNING REPRESENTATIONS

There are many ways in which Neural Networks can be used to learn representations. The most prevalent is by applying operations that transform the original data, mapping it into potentially more abstract features (BENGIO; COURVILLE; VINCENT, 2013) which can be more easily classified or regressed in the output layer. This mapping is generated by the hidden layers and is inherent to all Deep Learning models. These representations produced can be used in another downstream task or domain. This practice is known as Transfer Learning and has been successfully employed in tasks such as text classification (DEVLIN et al., 2018), fingerprint liveness detection (NOGUEIRA; De Alencar Lotufo; Campos MacHado, 2016), emotion classification in videos (KNYAZEV et al., 2017).

Another way in which these representations can be learned is by explicitly training the model to learn these mappings, rather than learning them as a product of another task such as classification or regression. Representations also do not need to be learned by gradient optimization in a Neural Network, although the main focus of this work is on this particular type of representation. For the remainder of this work, we introduce several representation techniques and conclude by presenting some frameworks for learning representations that have been studied in the literature.

2.3.1 Extracting features from text

Machine Learning algorithms operate with numerical data to learn its parameters. To work with natural language text data, it is needed to transform the text into a representation the computers can process. One common way of representing words as numerical data is by a

n-dimensional vector, where n is the cardinality of the vocabulary, and the i-th dimension corresponding to the i-th word receives the value of 1 and all other dimensions are set to 0. This representation is known as one-hot encoding. By summing the vector-representation of each word in a document (e.g. the description of a vehicle listing), a fixed-size representation is generated. This representation is known as Bag of Words, which, effectively, is a histogram of word occurrences. In the binary version of the Bag-of-Words representation repeated words aren't taken into account multiple times, resulting in a vector where all values are either zero or one. This histogram of words (term frequencies) can also be normalized by a weighting factor that gives less importance to tokens that appear in a lot of documents and might not contribute with much information towards, for example, a classification task. One particular weighting scheme is known as the IDF (JONES, 1972), that penalizes frequent word and rewards rare words by a factor $\log \frac{n}{1+\mathrm{df}(t)}$ where n is the total number of documents and $df(t)$ is the number of documents where the token t occurs.

2.3.2 Sparse and Dense word representations

In Section 2.3.1, one-hot encoded representation of words and Bag of Words representation of documents were introduced. These are forms of sparse representation, because the encoding of documents, where only a small fraction of the words in the vocabulary appear, results in very sparse vectors. This encoding model has some undesired properties. One is the high dimensionality which can be inefficient or hampering during learning of models' parameters. The other undesired property is that words are orthogonal to each other, meaning there is no special relation or similarity between a pair of words in comparison to any other pair.

A more efficient representation would be a dense representation, where the semantics and statistics of words are encoded in the vector space (e.g. synonyms being closer than unrelated words). There are some ways of achieving such representations, such as Latent Dirichlet Allocation (BLEI; NG; JORDAN, 2003) for topic modeling or factorization of co-occurrence matrices with Singular Value Decomposition. The skip-gram (MIKOLOV, 2013) and continuous bag-of-words (MIKOLOV, 2013) are two Neural Network-based strategies of learning dense word representations by predicting a word given the words in a neighboring window or vice-versa. This effectively encodes the representation of words in an unsupervised manner.

Entity embeddings of categorical variables (GUO; BERKHAHN, 2016) are a generalization of the neural embedding mechanism. It acts equivalently as a look-up table (embedding matrix) where each entry is the embedding associated with one entity that is retrieved by matrix multiplication with a one-hot encoded vector. The weights are optimized as usual with the rest of the network in the same downstream task. This approach, however, does not benefit from the same pretraining mechanisms that word2vec (MIKOLOV et al., 2013) or GloVe (PENNINGTON; SOCHER; MANNING, 2014) do, where you can use the same corpus and syntactic statistics to learn an initial set of weights that will be further optimized for the task. That is because the possible categories are not co-occurring for the same variable. Still, using weights learned from other domains is a possibility.

Other extensions to embeddings add subword representations, such as character n-grams as done in fastText (JOULIN et al., 2017) or consider multiple types of entities, e.g. sentences, documents, users, and for various classification and ranking tasks, as done in StarSpace (WU et al., 2018).

2.3.3 Frameworks for learning

In this section, we discuss alternative methods for learning representations other than the learning that is already implicitly done by the hidden layers in DNNs when trained for regular tasks such as classification and regression. Although in the previous sections we have discussed not only how to represent but also how to learn the representations, we mostly focused on Natural Language applications and representing words, sentences or documents. In this section, the focus is on more general approaches of supervised, unsupervised and self-supervised neural learning frameworks for representations.

Autoencoders (BALLARD, 1987; RUMELHART; HINTON; WILLIAMS, 1987) are one of the earliest types of neural networks specifically designed to learn representations. The autoencoder framework optimizes its parameters trying to reconstruct its inputs. In stacked autoencoders, multiple layers are trained one at a time; the first layer tries to reconstruct the input features in the original space, while the subsequent layers are trained to reconstruct the latent representations from its preceding layer (FUKUSHIMA, 1980). After training a layer, its weights are frozen and no longer updated during the training of the subsequent layers. There is no need for training labels during this stage and as such the model benefits from being able to be pre-trained on a larger number of samples before being finetuned to another task by adding an output layer. Denoising autoencoders try to reconstruct the original input from a noisy version of it (GOODFELLOW; BENGIO; COURVILLE, 2015). There are other extensions of autoencoders, but they are beyond the scope of this work.

Siamese Networks, originally introduced by Bromley et al. (1993), are a learning framework where two identical branches in a network are trained in a supervised manner to learn a representation of the input. The branches are constrained to have the same initial weight and are updated simultaneously. Effectively, they are a copy of the same subnetwork. The representation is commonly used for comparing both inputs and computing a distance measure. The original work uses it for signature verification. Subsequent works have applied it for facial verification (CHOPRA; HADSELL; LECUN, 1997). Siamese Networks are a method for metric learning, because its representations aim to be useful for distance comparisons and are a product of the learning objective (minimize the distance of similar pairs) rather than a side effect of it, e.g. minimize the regression error.

Triplet Networks (HOFFER; AILON, 2015) extends the Siamese Network by training a weight-tied three-way copy of the branch, where the training samples are not only pairs of similar or dissimilar instances, but rather a triple containing a similar and a dissimilar item for an anchor item which is being compared to the two. This framework encodes information about what is supposed to be similar or dissimilar at the same time. Such distinction is useful because the model implicitly learns when, for example, two different persons are similar (e.g. in the context of object detection when they both are of the class "human") and when they are dissimilar, such as in the context of person identification. This network encodes a pair of distances between the reference instance and the positive and negative instances.

Following this trend, the Triplet Loss was introduced in Schroff, Kalenichenko and Philbin (2015). Rather than achieving metric learning by relying on architectural design, the authors propose a loss function that minimizes the distance between the anchor instance and the similar instance, while maximizing its distance to the dissimilar instance. Their method learns an embedding in a Euclidean space and is evaluated in the task of face verification. The embedding of two faces can be directly compared to decide if they are similar because the loss is relevant to the task. In contrast, transfer learning using the bottleneck features of a CNN for this task requires fine-tuning.

In a vein similar to autoencoders, Correlational Neural Networks (CHANDAR et al., 2016) learn to reconstruct one view of the data from another view.

On self-supervised approaches, where the model is trained by exploiting characteristics of the data to supervise the learning process. One such example, in the task of Language Modeling, is to predict the next word given a sequence of words. In Radford, Jozefowicz and

Sutskever (2016), by training a character-level recurrent model to predict the next character in the sequence over a large corpus of user reviews, the learned representations were shown to be useful for text classification tasks in different domains. More interestingly, they discover a single neuron that is highly correlated with the polarity of the sentiment associated with a character sequence and that, by fixing its output in the recurrent network, the sentiment in the review was also affected.

In Bengio et al. (2003), a neural language model learns distributed representations for words that outperformed the previous state-of-the-art n-gram based models. The word2vec (MIKOLOV et al., 2013) model extends this framework with two strategies: (i) continuous bag of words model (CBOW), which frames the self-supervised problem of learning a word based on a context (a window of neighboring words around it), (ii) and the continuous skip-gram model (Skip-gram), where the model learns to predicts the surrounding words given the current word. It was further improved with negative sampling (MIKOLOV et al., 2013), to sample frequent words less often.

By training a language model on a corpus and then fine-tuning for classification tasks, ULMFiT (HOWARD; RUDER, 2018) learns representations for documents in a self-supervised manner. Following, derivative works using bidirectional RNNs for the language model and extracting a weighted combination of the internal states of every layer of the language model (PETERS et al., 2018), using Transformer networks (RADFORD; SALIMANS, 2018) and training for two simultaneous tasks in Devlin et al. (2018): 1) masked LM, which allows bidirectional models to learn to predict the next word without "cheating", as they would otherwise already have access to the next word, due the the bidirectionality. Since bidirectional LMs already have access to the next word in a multi-layered context, predicting the next word is simple. And 2) next sentence prediction, where the model receives sentence pairs and predicts whether one sentence follows the other. This task improves the embeddings for tasks where a relation between two sentences is modeled, such as in QA and Natural Language Inference.

3 RELATED WORK

In this section, we discuss related studies and present a comparison to the proposed solution in this work. To select relevant works, we used the following criteria:

- The main objective is to predict the price of products or learn representations;
- The solution employs some form of multi-view learning;
- The work publication is within 5 years of the current year.

We further subdivide this section into three major groups of related studies: the ones where the focus lies on some form of text-based regression, the ones where the focus is on representation learning and finally the studies which employ some form of co-attention.

3.1 TEXT REGRESSION

In Bitvai and Cohn (2015b) the task of loan rate prediction is tackled using Gaussian process-based models. The method proposed in the study handles both structured and unstructured data. From the unstructured features, TF-IDF histograms and LDA topics are extracted.

In Tay et al. (2018), they study the application of neural networks for the task of automatic text scoring (ATS), which are tools used for evaluation of tests which contain text-based answers such as GRE or TOEFL. They augment a recurrent model with neural coherence features, which are a form of shortcut connections that expose hidden states to deeper layers further in the network, which they state alleviates the vanishing gradient problem.

In Dereli and Saraclar (2019), the authors employ the CNN architecture introduced by Kim (2014) in the context of predicting stock return volatility, a measure on the spread of daily closing prices of a stock that quantifies financial risk. They use the text present in reports to predict the associated volatility value 12 months after the report was published.

In Xiong, Zhong and Socher (2019), the authors propose a co-attention encoder-decoder network for the task of question answering. The network predicts a span where the answer is contained. While this model is not specifically a text-regression model, their co-attention model was the basis for one of the network architectures proposed in this study.

In a vein similar to this study, the authors in Subramanian, Baldwin and Cohn (2018) use the CNN model for text regression, instead of the originally proposed text classification. They employ it in the context of predicting the popularity of online petitions based on its textual content. Alongside the feature map generated by the CNN model, the authors also include custom hand-engineered features such as the ratio of indefinite and definite articles, the ratio of subjective words, and other syntactic features such as count of nouns, adjectives, and adverbs.

In the work proposed by Paetzold and Specia (2018), a CNN model is also enriched with hand-engineered features. Additionally, they perform the training with a multi-task objective function. Their reported results show that although the overall solution did achieve state-of-the-arts results, it is less computationally heavy and that the multi-task learning needs to be carefully designed, otherwise it might compromise the model performance.

Another work where text regression is performed is in Bitvai and Cohn (2015a). The authors take the problem of predicting the box office revenue of movies based on reviews by movie critics and structured attributes about the movie. The architecture is based on CNNs for text, only with a real-valued output for the task of regression. Metadata about the movie is concatenated with feature maps after the pooling operation, and domain information is integrated as a one-hot encoded vector appended to each n-gram feature, signaling the source site for the review. According to the reported results, one of the main sources of improvements comes from adding non-linear activation functions, which significantly increases the performance when compared to the simple linear regression baseline models. Introducing meta-information about the movie (the structured attributes) and domain information (source site) at the same time shows no significant improvement, possibly due to its complementary nature. Furthermore, the authors introduce a model interpretation strategy that helps to find the highest-impact n-gram features for the final predictions.

3.2 REPRESENTATION LEARNING

In Pham, Kruszewski and Boleda (2016), CNNs - more commonly used in Computer Vision related tasks - are explored in the task of Language Modeling. Although CNNs have been previously used in other NLP tasks, such as Text Classification, its use for sequential prediction tasks were largely unexplored. The authors introduce two CNN-based architectures, one being comprised of concatenated feature maps of different kernel sizes, achieving higher granularity of context (3-gram and 5-gram), and show comparable results to similar-sized state-of-the-art recurrent models.

In Hoffer and Ailon (2015), authors introduce an architecture which explicitly learns a similarity measure between examples, by maximizing the distance measure between examples of the same class and minimizing the distance to an anchor example of a different class. Despite its advantages compared to Siamese Networks, such as not requiring a precise calibration for the context of similarity (e.g., person vs other objects or person vs other individuals), more recent techniques on metric learning rely more on loss functions, rather than architectural characteristics.

In the context of Common Representation Learning, Chandar et al. (2016) introduces an AE-based approach to learn a joint representation of different views of the same data. The proposed method (CorrNet) learns a joint representation by learning to reconstruct each view of the data independently, and to reconstruct one view from the other, explicitly maximizing the correlation between the representations.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos (2017) propose a model to generate sentence embeddings with a bidirectional LSTM. They encode the word embeddings with the recurrent network and reduce the hidden state matrix (which has the same size as the input sequence) to a fixed-length matrix by computing self-attention weights with a two-layer Multi-layer Perceptron. Each row of the final attention matrix corresponds to a set of weights that highlight some interaction between the representations across each hidden state. They enforce diversity in those weights by introducing a regularization term that is added to the loss and minimized together with the original loss. They evaluate their method on three tasks: author profiling (age classification from tweets), sentiment analysis and textual entailment.

For learning representations of sentences, the authors in Subramanian et al. (2018) explore a single recurrent encoder without attention mechanisms in a multi-objective training setting. Five out of the six tasks are formulated as sequence-to-sequence problems such as machine translation, where the output is the same sequence but in another language. The other task is text classification. As for the multi-task setup, they employ a simple approach of performing a parameter update then uniformly sampling another mini-batch from other tasks. The reported results show that the model learns representations that are as good or superior to other general-purpose sentence representation methods.

The authors of Meng and Li (2016) propose a multi-modal similarity metric that matches tokens in two input sequences that are different but have a similar meaning in the task of answer selection. The remainder of the architecture is a combination of convolutions and pooling layers over the intermediate mapping generated by the similarity metric, before concatenating with features that measure word overlap count between the two input sequences. The network is optimized with pointwise ranking loss, which only looks at one answer at a time and predicts the relevance of the candidate answer as the actual answer for the question, essentially a classification task. Additionally, they use the Frobenius norm as regularization for the parameters that compute the similarity matrix. The reported results show improvements over previous state-of-the-art approaches.

3.3 CO-ATTENTION

Xiong, Zhong and Socher (2019) use a co-attention mechanism for the task of Question Answering by predicting the answer span. Because the predicted spans are numerical outputs, this model can be considered a special type of regression. The proposed network uses co-attention for learning a co-dependent representation of the question and the document. Additionally, they also introduce a dynamic pointing decoder that is based on an LSTM. One interesting aspect of their co-attention encoding is that, based on Cui et al. (2017), after computing the attention scores for the summaries, instead of computing them separately, the document summary also includes an attended version of the summary of the question (that was already attended).

In Rao et al. (2019), the authors propose a model to tackle a series of tasks relating to similarity modeling between two short text snippets (context and query). Their model uses co-attention to model semantic matching between the sequences, learning a query-aware representation for the context as well as a weighted average of the most important words in the context in light of the query (summarized context). They generate a final enhanced representation by concatenating raw representations of the document, the attended (contextualized) document, and tensor-products between raw context and attended context, and between summarized context and attended context. They use an LSTM to encode this enhanced representation and the last hidden state is used as the output for the semantic matching branch in their proposed model.

3.4 SUMMARY

Overall, most solutions that use structured features, only do so with a simple concatenation of the structured and unstructured features. In our proposed solution, the interactions between both are also modeled, so it is a more general framework. From the perspective of Representation Learning, there are many relevant approaches: generative models, self-supervised alternatives, multi-task objectives. In this particular study, we focused on the learning of representation by specifically targeting one downstream task (price prediction). Following is presented a summary of related text regression and co-attention works and key aspects in which they differ from the proposed method.

Bitvai and Cohn (2015b): Scope: Regression, loan rate prediction with Gaussian process-based models. Uses structured and unstructured data. **Key differences:** Does not encode relations between modalities. Uses only count-based features such as TF-IDF and LDA.

Tay et al. (2018): Scope: Automatic text scoring (grading text-based answers) using Neural Networks. Includes a form of relation features computed from tokens in the text. **Key differences:** Does not consider structured data. Utilizes recurrent models which are more computationally expensive than our feedforward architecture that only relies on embeddings to perform similarity computations.

Dereli and Saraclar (2019): Scope: Predicting a measure of stock volatility that quantifies financial risk from stock reports (text). **Key differences:** Does not encode any form of relations between modalities to enrich representations and only uses unstructured data.

Xiong, Zhong and Socher (2019): Scope: Question answering with a co-attention encoder-decoder network. Encodes relations between two textual inputs (question and document) to provide the span of the answer. **Key differences:** Although it is not particularly applicable for the task addressed in the study, they do not utilize structured data. Additionally, our approach encodes relations between slightly different textual modalities as well, categorical variables and unstructured text, rather than both being unstructured text.

Subramanian, Baldwin and Cohn (2018): Scope: Online petition popularity prediction based on its textual content. Uses hand-engineered features to enrich the feature space for the task. **Key differences:** Their approach does not take advantage of relations between modalities present in the data (such as title and description or keywords and description). Moreover, they make use of hand-engineered features that require more work and domain knowledge to successfully employ.

Paetzold and Specia (2018): Scope: Exploits character-level information and hand-engineered features to perform text regression in two tasks: quality estimation of machine-translated text and emotion intensity analysis (quantifying a certain emotion in a text), using deep CNNs and in a multi-task setting. **Key differences:** Although the hand-engineered features are positivity scores extracted from a treebank of annotated sequences, i.e they do not require additional manual labeling, those features do not encode very complex relations between the modalities.

Bitvai and Cohn (2015a): Scope: Box-office revenue prediction from reviews (text) by movie critics and structured attributes of the movie (genre, director, actors). **Key differences:** Their model is trained with structured and unstructured input. However, complementary information is not explicitly learned by the model.

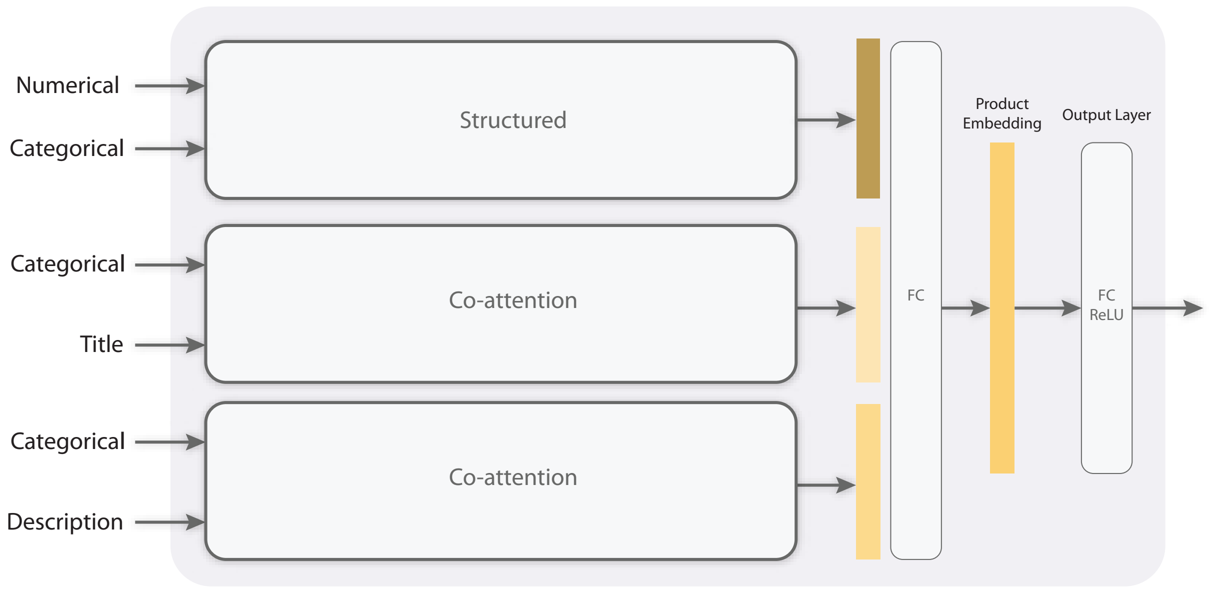
Rao et al. (2019): Scope: Similarity between short text snippets (classification). A co-attention network models a semantic matching between the sequences, as well as a co-dependent representation of each snippet in the pair. **Key differences:** Regarding the co-attention component of the network, after computing co-dependent representations, their model uses a recurrent network to produce final contextual embeddings. Our proposed method uses a simple feedforward network.

4 PROPOSED NETWORK

In this chapter, we present our proposed attention-based network that generates product representation from structured and unstructured information to perform product price prediction.

As aforementioned, we consider in this work two different views commonly present in product offers: specification (structure information), which describes the main attributes of the product, and text (unstructured information) about the product present in the title and the description of the offer. To jointly learn patterns on those different views to perform price prediction, we propose a three-branch neural network that deals with the structured view (structured branch) and two co-attention branches for the product's title and description that simultaneously learns patterns on the text and structured view. More specifically, we apply a Multi-layer Perceptron Network (MLP) on the structured view, and a co-attention network to attend more relevant parts of the text regarding the structural attributes and vice-versa, and capture information in the text complementary to the structured attributes. Each one of those branches produces a distinct representation of the product in form of a vector. These vector representations are then concatenated and fed into a fully-connected neural network, which provides the input to the regression layer (a single neuron with a RELU activation function). The output of the hidden layer embeds the learned patterns of the different views for the price prediction task: the product embedding. Figure 3 presents an overview of our proposed network. In the remaining of this section, we give further details about the branches that compose the network.

Figure 3 – Proposed solution.
Source: Authors' own elaboration.



4.1 STRUCTURED BRANCH

Structured tabular information is one of the most common types of data used in Machine Learning systems. While Deep Learning usually employs the raw data itself for learning the models' parameters (e.g. an image of an object instead of its measurements), the data available in structured form might contain useful information which can be used directly as input for the model or used to leverage information in other modalities. In this work we apply a standard MLP network to model the structured data as we further detail its implementation in this section. Our experiments showed that a simple feedforward network for this modality was sufficient, and that larger architectures were less stable to train, possibly due to the lower dimensionality of data (2 numerical variables and 3 categorical variables with a rather small vocabulary of words in comparison to the description of a listing, for example).

In our context, the structured information are the attributes of a product. We consider them for the price prediction task since they usually have some influence in the product price. For instance, the size of a TV screen or the brand of a cellphone has some effect on the price of those products. Product's attributes are usually composed of numerical and categorical variables.

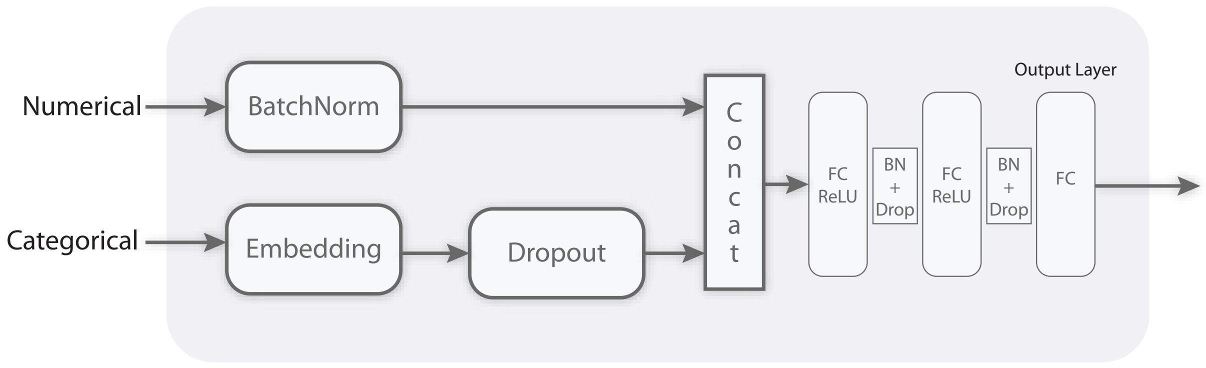
A common strategy for normalization of numerical attributes is to use min-max or z-score on the the input data. In this work, we opted to use Batch Normalization (IOFFE; SZEGEDY, 2015) since, in our preliminary experiments, it presented better results than the traditional normalization techniques. With Batch Normalization (BN), inputs are normalized to achieve zero mean and unit variance in each batch. The BN layer also has two learnable parameters (γ and β) that allow the training to counteract the normalization by changing only these weights, avoiding the network to lose stability by changing all the weights. In Equation (4.1) we show the main equations that compose BN. The first equation is the transformation applied to the inputs of the BN layer (x_B), scaling it according the mini-batch mean (μ_B) and standard deviation (σ_B) statistics. The second one is the result of the final transformation with the learnable parameters γ and β.

$$\begin{aligned} \hat{x}_B &= \frac{x_B - \mu_B}{\sigma_B}, \\ \text{BN}(x) &= \gamma \hat{x}_B + \beta \end{aligned} \tag{4.1}$$

As for categorical variables, each of their values need to be associated to a number in order to be processed by the MLP model. A common approach to deal with categorical types in predictive models is using one-hot encoding (GENTZKOW; KELLY; TADDY, 2019). This strategy, however, can produce a high number of features in the case a category with high cardinality (large number of values), and it treats the category's values completely independent, ignoring possible relations between them. To deal with that, we use Entity embeddings (GUO; BERKHAHN, 2016). Entity embeddings map sparse the one-hot encoded inputs to a dense, lower dimensionality. Effectively, this is equivalent to a linear mapping layer for each categorical variable, before feeding into the subsequent layers of the model. The layers' weights, i.e. embeddings, are randomly initialized and are optimized for the same task along with the rest of the network. We also use Dropout (SRIVASTAVA et al., 2014) with a small probability (0.04, which showed good results on our preliminary experiments) after the embedding look-up, because the embeddings are also optimized and can overfit, therefore some form of regularization is desired (GAL; GHAHRAMANI, 2016).

Finally, the two intermediate representations from the numerical and categorical attributes of the product are concatenated and successively passed through a series of fully-connected linear layers (FC) with ReLU activation (GLOROT; BORDES; BENGIO, 2011) and batch normalization (BN) with dropout regularization layers (Drop). Figure 4 presents the components of the structured model.

Figure 4 – Structured model.
Source: Authors' own elaboration.



4.2 CO-ATTENTION BRANCH

As mentioned before, the text in the description or title of a product listing might contain additional information not present in the structured part that can improve the quality of the price prediction. At the same time, much of the text present might not be relevant to the price estimation at all. A reasonably safe assumption is that some sentences are more important than others and that in general, they appear in the vicinity of other relevant sentences, i.e. some paragraphs are reserved for further detailing the product while others might focus on other aspects which do not affect the price as much, such as information about the car dealership.

Previously, co-attention networks have been used primarily for matching problems such as question answering (XIONG; ZHONG; SOCHER, 2019) and text similarity (RAO et al., 2019). In this work, we use co-attention in a different way: we aim to combine structured attributes and textual information by jointly handling these modalities and their interactions. The reasoning behind this is that the co-attention network can lean how to leverage interactions between the structured and text modalities to improve the overall quality of the model by attending more relevant parts of the text according to the structured attributes of the products and vice-versa, and also learning complementary patterns in the text useful for our task.

Since we try to leverage text in the product offer in two different fields: title and description, the co-attention branch is composed of two sub-branches: one to model the interaction between the title and the categorical attributes, and another one for description and the categorical attributes.

While numerical attributes are also likely to appear in the description body or in the title, they are harder to embed because a simple tokenization of the text leads to many unique values, for which the co-occurrence statistics with other words are very low. This leads to their embeddings being optimized very few times and result in poor representation. If quantized, e.g. by rounding to the nearest thousand, their representations might see some improvement because the tokens will become more frequent. For example, there might be only one car listing with mileage equal to 52146, but many cars with about 50K miles registered in the odometer. Another problem is accidental matching. Numerical attributes might be similar and, when quantized, assume the same value. This will signal an unintended strong relationship between the tokens. Additionally, some listings have information such as "this car is 22000 miles below the market average", without specifying the actual odometer reading. This adds more accidental matching possibilities when considering the numerical attributes. Lastly, we preprocess the text as to not leak the target variable, the price, in the listing description. In some cases, it might also accidentally remove one of the other numerical attributes although we specifically look for patterns including the dollar sign. As such, there are many problems when considering numerical attributes and they need to be tackled carefully before being considered in the co-attention.

The inputs of each sub-branch are the categorical attributes of the products and the text (title or description), which are passed to an embedding layer. This layer has a shared vocabulary between all the inputs. The textual representation is mapped to an embedding space, transforming each sequence of words or characters into a sequence of vectors in that space. Similarly to the structured branch described in Section 4.1, categorical variables are treated as single-token, even though sometimes they are made of multiple words (e.g. Alfa Romeo and Rolls Royce). The inputs are numericalized as usual, via tokenization and one-hot encoding, and then embedded into the same space.

The embedding representation of each categorical variable is concatenated into a single matrix C, similar to the sequence of words in the text into a matrix T. An important difference, however, is that the categorical matrix is not a real sequence. An affinity matrix $L = CT^\top$ computes the similarity of each token pair between the categorical variables and the sequence of words in the text. Since the tokens from each modality are embedded in the same space, similar vectors will result in a high similarity coefficient. To compute the attention scores, the affinity matrix L is normalized row-wise for text weights:

$$A^T = \mathrm{Softmax}(L) \tag{4.2}$$

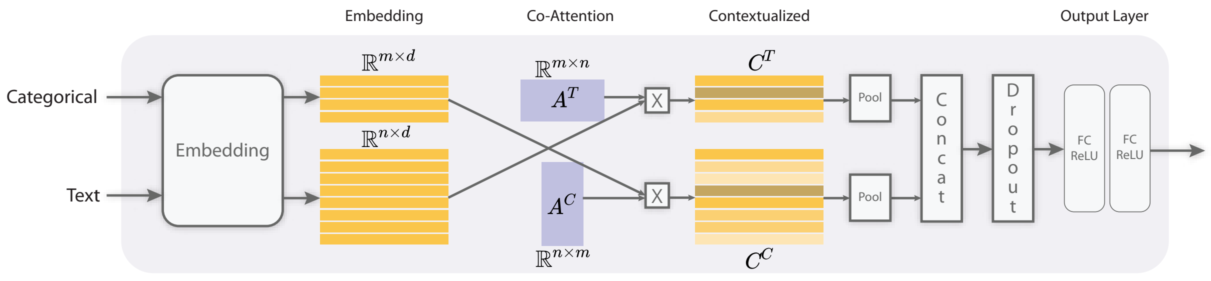
And column-wise for categorical weights:

$$A^C = \mathrm{Softmax}(L^\top) \tag{4.3}$$

These weights are used to generate the context of categorical variables taking into account the text words $C^T = CA^C$, and the context of text words taking the categorical variables into account $C^C = TA^T$. These generate the enhanced contextualized representations. These representations are still of variable length. In order to reduce this to a fixed size vector, we use a pooling operation. We use mean-pooling since in our preliminary experiments it showed the best performance. Finally, from the concatenation of the two resulting representations, which now have a fixed size, the model generates the predictions by using an MLP. We use a two-layer feed-forward network with ReLU activation functions, and apply dropout on its inputs (the concatenated representation) as a means of regularization.

Since the attention mechanism has no learnable parameters, all the predictive power resides in how much information the word embeddings (co-attention's input) and the MLP (co-attention's output) can learn.

Figure 5 – Co-Attention model. There are two instances of this model in the final architecture, one using the title and another using the description for the text sequence input. **Source:** Authors' own elaboration.



4.3 MODEL TRAINING

To train the network, we use the price prediction task (regression) to compute the Mean Squared Error loss and update the model's parameters. Other loss functions are also suited for the regression task, such as the Huber loss (HUBER, 1964). However, our experiments have shown no significant difference between the two. The price variable, however, does not behave normally-distributed. For that, we use the natural logarithm of the actual price as the target, which rescales the distribution, improving the training performance. The loss is minimized by stochastic gradient descent, with backpropagation for gradient computation and Adam weight-update scheme (KINGMA; BA, 2015). In our experiments, using an adaptive algorithm was better suited in this particular domain. We have also experimented with a regularized variant of the Adam algorithm that employs weight-decay, the AdamW (LOSHCHILOV; HUTTER, 2019) optimizer, but there was no significant difference in observed performance.

5 EXPERIMENTS

In this chapter, we describe the experiments conducted to validate the proposed solution. We include comparisons with baseline classical learning algorithms. Furthermore, we include ablation studies to provide further insights into the importance of each branch in the complete attention-based architecture. Later, we present and discuss the results and their implications.

5.1 EXPERIMENTAL SETUP

5.1.1 Data

The dataset used in the experiment consists of 57285 records that were scraped from 11 car listings USA websites. Table 1 presents the websistes and how many car offers, we collected from each one of them. The following fields are common among all sources: year, mileage, price, manufacturer, model, title, location and description. In Figure 6, we show the distribution of the year of the cars in our dataset. As one can see, there is a great variability in distribution: most of the cars are from recent years although there are also old cars, for instance, from 1950's, 1960's and 1970's.

Regarding mileage, there is a reasonable proportion of new cars (mileage equals to 0) and used cars as shown in Figure 7.

Table 1 – Number of collected listings from each source website. In total, there are 11 sources and 57285 records.

Source	Count
sellmycar	23556
usaa	9983
truecar	9579
classiccars	5035
autolocator	3662
hemmings	1596
bestcarfinder	1580
kbb	1209
ksl	446
omaha	387
oodle	252

Figure 6 – Year distribution.
Source: Authors' own elaboration.

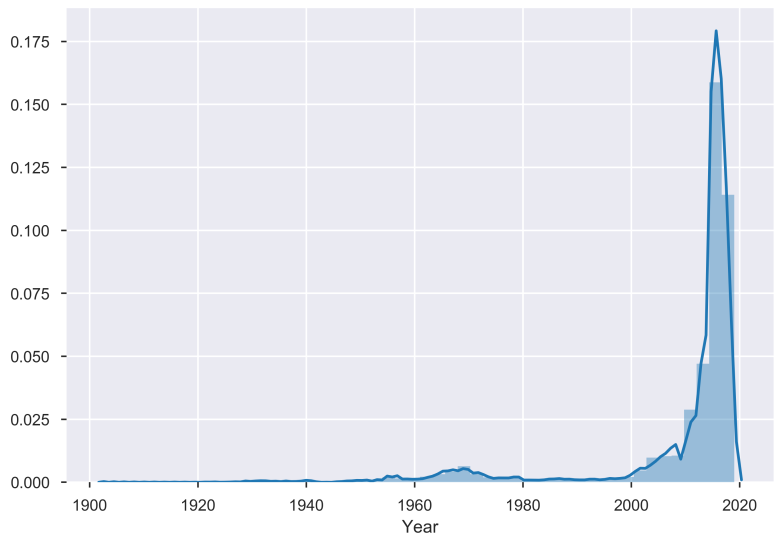
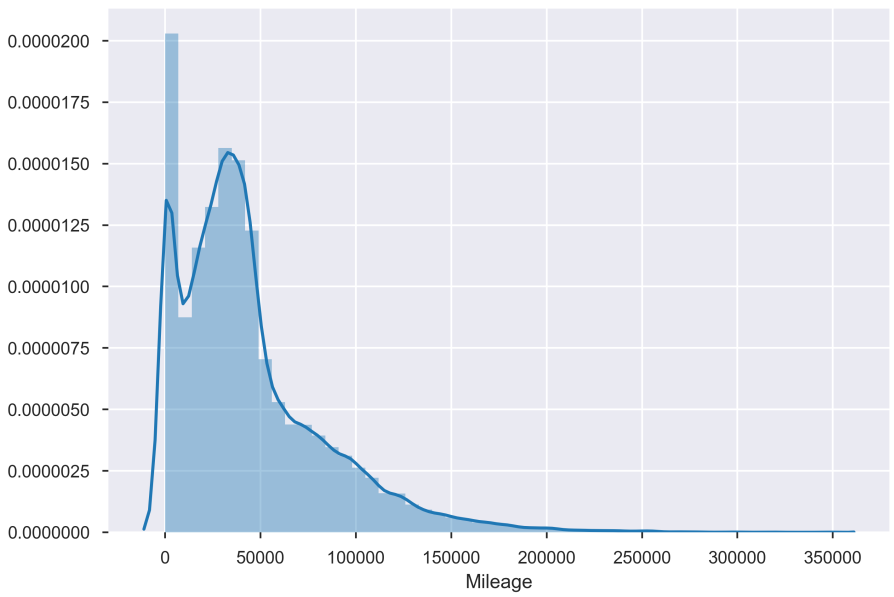


Figure 7 – Mileage distribution.
Source: Authors' own elaboration.



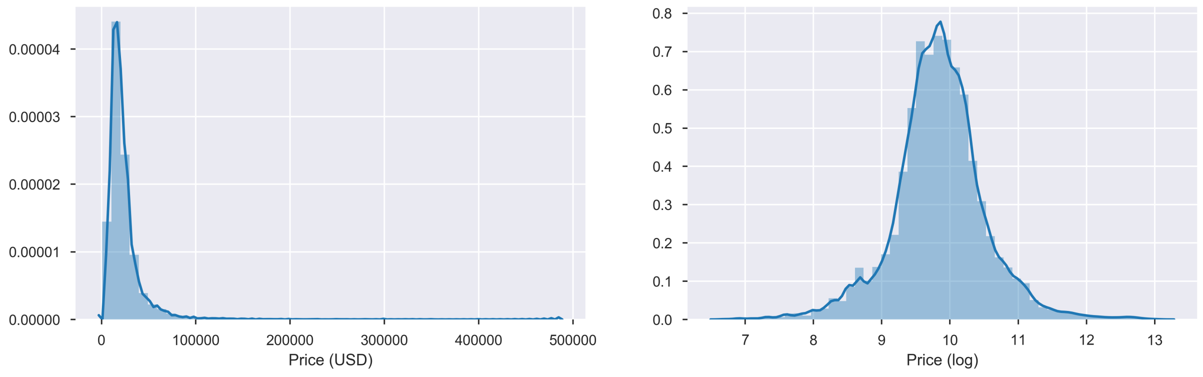
From the location field, which usually contains complete addresses and sometimes phone numbers, the corresponding US state was extracted. Of those variables, title and description are considered to be unstructured. For the scraping process, the text field containing sellers' comments and further details on the listing were considered as the description body. Table 2 provides a sample of 3 entries in our dataset with their respective values.

Table 2 – Three samples from the data. Title and descriptions were trimmed. The last column, price, is the target attribute.

Year	Mileage	Manufacturer	Model	State	Description	Title	Price
2016	28385	mercedes-benz	a	OH	Included Features Control Code, KEYLESS GO Pac...	2016 Mercedes-Benz E-Class E 350 4MATIC Luxury...	31363
2011	88763	hyundai	genesis	CA	Just bought a Prius and don't have room for th...	2011 Hyundai Genesis	9800
2008	45080	mazda	miata	CA	Our 2008 Mazda MX-5 Miata Convertible presente...	Used 2008 MAZDA MX-5 Miata SV 2dr Convertible	10398

The price (the target variable) was converted to float and thousands separators and dollar signs were removed. As mentioned in Section 4.3, we apply a log transformation on the price to turn its distribution close to a normally-distributed one as the histograms in Figure 8 depict.

Figure 8 – Histograms of price distribution in (left) normal USD scale and (right) log-scale. **Source:** Authors' own elaboration.

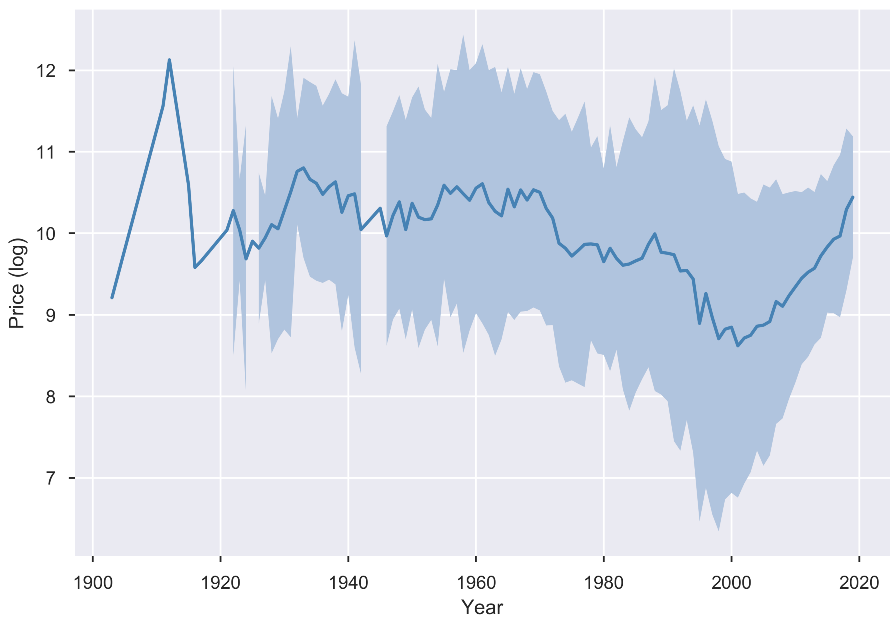


The data was cleaned to remove: (1) the top and bottom 0.5% outliers according to price and mileage; (2) samples with missing values that could not be imputed (missing title, description, location, mileage or price); and target leakage in other variables, such as product description containing the price in its body. We perform a stratified split into 70% training and 30% test sets, maintaining the proportions of samples per source across the splits. For the Neural Network models, the training set is further subdivided into 30% validation which is used to perform model selection: (1) to determine the best snapshot (epoch) of the model's parameters after the training procedure is finished; and (2) also for hyperparameter tuning.

Since this a real-world dataset, it presents many challenges: there are multiple sources of listings, where entries might be posted freely or need reviewing before being listed; the title of the entry might follow different conventions in different sources; there might be considerably different writing styles or type of information included in the comments; and the source site might have listings with prices that are consistently higher or lower than the price in other sources.

A final observation about the dataset is that one of the sources is specialized in classic automobiles (Classic Cars). This breaks the pattern that older cars are usually cheaper, which adds further complexity to the data. In Figure 9 it is possible to observe that prices decrease with age, up to a point (around 2000s) where it starts going back up again. Therefore, some old cars might be of very limited availability, which increases its value, but it can also be an age-worn car with lower price. Adding the source site as a feature would likely increase the model performance, but we want a more general, source-agnostic representation. For this reason, we did not include this feature.

Figure 9 – Per-year average price of listings (line) and 95% confidence intervals (shade). No confidence interval is present when there is only a single listing for a specific year. **Source:** Authors' own elaboration.



5.1.2 Regressors

We assess the performance of the following algorithms for comparison with the proposed solution.

- **Linear Regression:** Linear Regression (LR) models are the simplest baseline in regression tasks. For this algorithm we utilize its R implementation (R Core Team, 2020) with its default parameters;
- **Random Forest** (BREIMAN, 2001): Random Forests are ensembles of tree-based models trained on subsets of the dataset's features and samples that can be used for classification or regression. We utilize the scikit-learn's implementation with varying number of models in the ensemble: [50, 100, 200, 700, 1000].
- **Lightgbm** (KE et al., 2017): is a Gradient Boosting decision tree-based algorithm where models learn and adjust the errors of other models in the ensemble. For the search parameters, we use number of trees: [50, 100, 200]; number of leaves: [3, 5, 4, 100, 300] and learning rate: [0.03, 0.05, 0.07, 0.1].
- **Support Vector Regression** (CHANG; LIN, 2011): We use the maximum-margin SVM regressor wrapper for libsvm available in the scikit-learn package with a linear kernel and varying regularization coefficients: [1, 5, 10, 100, 1000].
- **H2O AutoML** (AIELLO et al., 2015): is an automated machine learning toolkit that performs algorithm selection and hyperparameter tuning and builds ensemble of predictors. Those ensembles are composed of individual regression models with a weight associated to each one of them.

5.1.3 Feature sets

We evaluate the models in this section using the following feature sets:

- **Structured (STR):** the structured features of the car listings are divided into two sets: categorical: manufacturer, model and US state; and numeric: year and mileage. We represent the categorical features using one hot encoding;

Table 3 – Model hyperparameter settings used in the gridsearch. Best configuration in bold.

Branch	Hyperparameter	Values
All	Branch output size	64, **128**
	Embedding	64, **glove100**
Co-attention	Dropout	0.5, **0.1**

- **Unstructured (TXT):** the unstructured features of the car listings: title, extracted from the title HTML tag, and description. Since the training time of some regressors are influenced by the number of features, we removed stopwords and selected the 4,500 most frequent words to compose this feature set.
- **Combined (STR+TXT):** the combination of both structured and unstructured feature sets;
- **Car embedding (CE):** the intermediate features generated by the proposed neural network architecture, immediately before being passed to the output layer that generates the price predictions.

5.1.4 Model settings

The proposed architecture was implemented using the PyTorch framework (STEINER et al., 2019). For model tuning, we adopted the grid search strategy with the hyperparameter settings outlined in Table 3. The number of values for each hyperparameter was limited due to the long training time of each configuration. For each configuration of hyperparameters, we trained the model for 100 epochs with no early-stopping mechanisms, and selected the one that displayed the lowest validation error. The embeddings are randomly initialized when not using a pre-trained embedding. When using pre-trained embeddings, the representation of tokens in the embedding layer is initialized when applicable, i.e. when performing word-level tokenization and words are known. After being initialized, the weights are optimized as usual, instead of being frozen. We create mini-batches by batching together sequences of similar sizes, using the BucketIterator available in the torchtext[1] library.

[1] https://torchtext.readthedocs.io/

5.1.5 Evaluation metrics

We evaluate the models by computing the mean squared error (MSE) and mean absolute log error (MALE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\,, \tag{5.1}$$

$$\text{MALE} = \frac{1}{N}\sum_{i=1}^{N}|log(y_i) - log(\hat{y}_i)|\,, \tag{5.2}$$

where y_i is the target price of product i, $\hat{y}_i$ is the price as predicted by the model, and N the number of products in the evaluation set.

Note that MSE (Equation (5.1)) is the cost function that is minimized during the training process. The model is trained to predict prices in log-scale, but all metrics with the exception of MALE are shown in the original scale. We also show the square-root of the MSE, i.e RMSE, which is more interpretable.

5.2 EVALUATION AND RESULTS

In Table 4, we report the performance of the baseline models, trained with different combinations of raw features, and their performance using only the features generated by our model, the car embedding. Each model had their relevant hyperparameters tuned (e.g. C for SVR, Number of trees and Max depth for Random Forest, etc) with grid search and the best setting was used to report the test set metrics.

For all regressors, the models trained using the CE feature set outperformed the same regressors with raw features (STR, TXT and STR+TXT). In fact, Random Forest using CE obtained the best overall result: MALE=0.117 and RMSE = 12302. Although the resulting regressor (output layer of the NN) of the proposed method does not achieve the best overall result, it is clear that the CE is the best feature set across all models, and that it is the most relevant part of the model.

Moreover, in almost all cases, LightGBM being the only exception, the car embedding is the best contender by a large margin. These results confirm that the learned car embedding encapsulates additional information that is not present when naively combining structured and unstructured features. This is evident, for instance, for SVR, which MALE was 0.185 on the

Table 4 – Regressors evaluated on the feature sets.

Model	Features	RMSE	MALE
Linear Regression	STR	15288	0.231
	TXT	39801	0.217
	STR+TXT	18555	0.171
	CE	12926	0.125
SVR	STR	15410	0.228
	TXT	31140	0.249
	STR+TXT	17672	0.185
	CE	13076	0.127
Random Forest	STR	13940	0.179
	TXT	16909	0.193
	STR+TXT	15440	0.168
	CE	**12302**	**0.117**
LightGBM	STR	14929	0.185
	TXT	14601	0.177
	STR+TXT	13560	0.130
	CE	12305	0.120
H2O AutoML	STR	15351	0.258
	TXT	18644	0.283
	STR+TXT	20341	0.299
	CE	12439	0.118
Proposed	CE	13071	0.126

STR+TXT feature set and 0.127 using CE. For individual models, without considering our generated features, the best overall result score was LightGBM using both structured and unstructured data: MALE=0.13 and RMSE=13560. Those results were also close to the final result of the same algorithm when using the car embedding: MALE=0.12 and RMSE=12305. This can indicate that although this model is the best performer on the original feature space, our embedding enables other much simpler and faster-to-train models to achieve comparable or better results. A concrete example of that is when we look at the results of Linear Regression using CE. the MALE score, for instance, of this configuration is better than all regressors using raw features.

Impact of Branches. Concerning our proposed model, to validate the contribution of each branch we selectively remove branches from the complete architecture before training to compare the final performance against the complete model. For comparison purposes, we also include combinations with a textual branch that does not model interactions between text and other modalities. The textual branch is based on the CNN model described in Kim (2014), with four parallel one-dimensional convolutions across the time-channel (sequence length), with kernels of size 2, 3, 4 and 5. It was originally purposed for text classification. Here, however, we use it for a regression task. The four generated feature maps are max-pooled into 4 vectors of equal dimension. In our preliminary experiments, adding a multi-head attention layer at this stage showed improvements. After computing the attended vectors (self-attention), they are concatenated and fed to a two-layer feed-forward network with ReLU activations.

The results of the ablation experiments are shown in Table 5. Considering individual performances, the co-attention branch has the lowest MALE error (0.151). The textual has a comparable although slightly worse error with 0.163, and the structured branch achieves 0.197. In contrast to the results with our baseline regressors, the performances using only structured features are comparable, but our network-based models have a much better performance when dealing with textual data. The individual performances when using only textual information, or textual and categorical information, are better than their structured counterparts. This leads to an interesting observation that although the more discriminant features are present in the structured data for classical learning algorithms, deep neural networks are better at extracting information from the textual data. Furthermore, DNNs do not fall short as an alternative when dealing only with structured features, it is only surpassed by tree ensemble models.

When inspecting the performance for pairs of branches, we can observe that all pairs outperform the single-branch alternatives. The pair with textual and co-attention branches, however, do not show significant improvement. We believe that this is because the textual branch does not add much information other than what the co-attention branch is already able to capture. In our proposed method, however, the combination of the structured information with the co-attention branch achieves a considerable improvement in performance, with 0.126 MALE. This combination achieves a better error than the pair of structured and textual branches. This corroborates with the surmise that our co-attentional model can enrich the representation of both modalities and produce a more relevant representation. The combination of the three branches, however, performs only slightly better than both of the pairs and performs worse than our proposed model. The redundancy between the branches and the added number of parameters of the textual branch may make the network harder to train and regularize effectively, which consequently hinders the performance of the system.

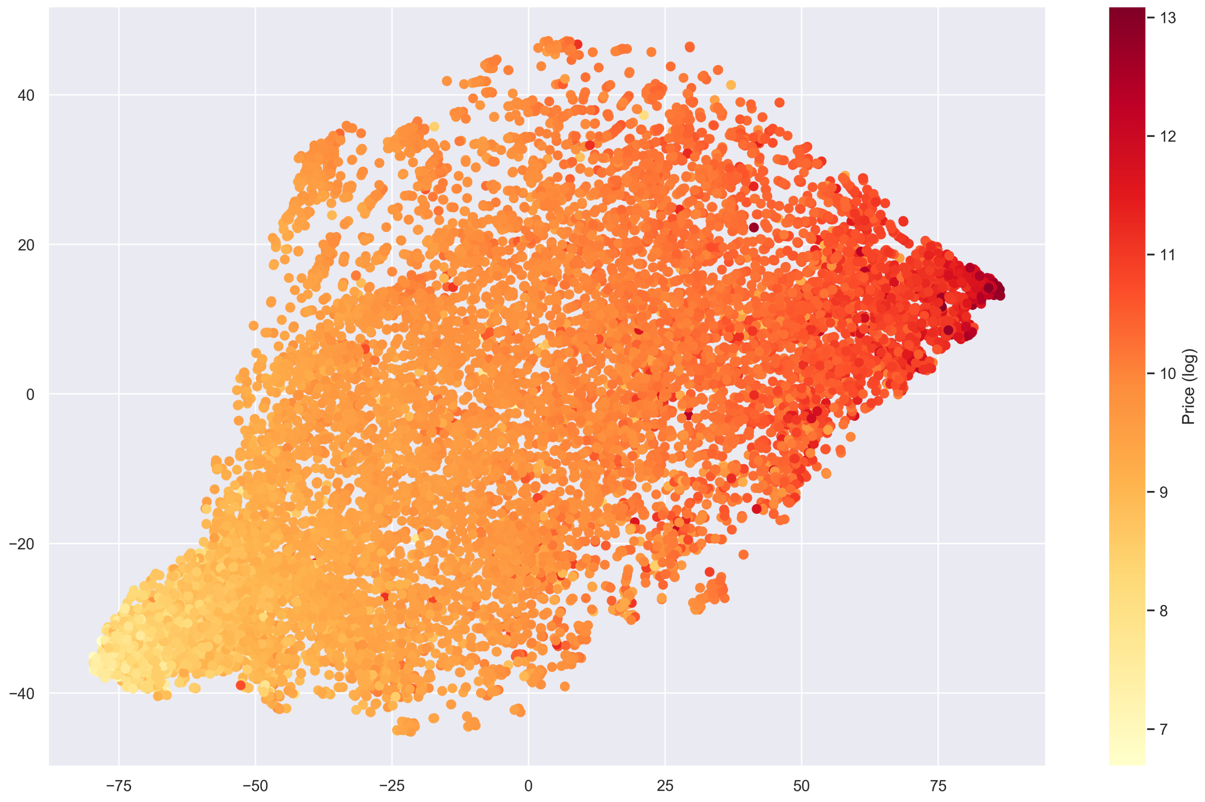
Table 5 – Results for the ablation experiments.

	Branch combination	RMSE	MALE
Singles	structured	15892	0.197
	textual	13616	0.163
	co-attention	13446	0.151
Pairs	structured, textual	12955	0.132
	structured, co-attention (proposed)	13071	0.126
	textual, co-attention	13159	0.148
Triples	structured, textual, coattention	13434	0.131

Representation Analysis. We discuss in this section how we assess the quality of the embeddings that were learned by the network while being trained to solve the price prediction task. In Figure 10, we display a two-dimensional projection of the records from the test set in original embedding space generated by using the t-SNE algorithm (NL; HINTON, 2008), color-coded by price. On the top and right-most part of the plot, we can see that there is a concentration of higher-price samples. In the opposite quadrant, the least expensive vehicles are more dominant. It is also possible to observe a gradient effect that, although with the presence of some outliers and a higher concentration around the mean values, indicates this unsupervised projection manages to extract the implicitly-encoded price information, and that similarly-priced samples are closer in this latent representation. This illustrates desirable properties of good representations according to Bengio et al. (BENGIO; COURVILLE; VINCENT, 2013) meaning that our embedding is able to form clusters (natural clustering property); and prices do not change drastically in the space of the embedding (smoothness property).

In the structured branch, there are also embeddings learned specifically for the categorical variables (the entity embeddings). We probed embeddings of car manufacturers to find the top 5 most similar to some query according to the Euclidean distance between their representations, and display the results in Table 6. For this analysis, we exclude manufacturers that have occurrence count lower than five, as they do not have the same representation quality of other manufacturers due to their associated parameters being updated much less frequently. Our results show that this embedding, situated very early in the network inside the structured branch, also encodes information about the task and can consistently map similar items close together. For instance, when queried for "Lamborghini", the manufacturer with the highest price average in our dataset, the closest points in the embedding space are manufacturers that

Figure 10 – Two-dimensional t-SNE projection of the learned embeddings for the samples in the test set. Better viewed in color.
Source: Authors' own elaboration.



also have a high-average price in the listings we collected, such as "Ferrari" and "Aston Martin". We expect that by using another objective function, or by employing multiple objectives, the embedding can capture other relationships between manufacturers other than having similar price ranges.

Table 6 – Querying the learned embeddings for car manufacturers for similar points in the embedded space.

Query	Top 5	Average Price (USD)
Lamborghini	-	271591
	Ferrari	137567
	Aston	119979
	Aston Martin	124978
	Hummer	17253
	Audi	23796
Rolls-royce	-	116298
	Ferrari	137567
	Bentley	87142
	Aston	119979
	Jaguar	23795
	Studebaker	30290
Volkswagen	-	12429
	Ford	19112
	Volvo	20431
	Porsche	51561
	Cadillac	20480
	Saturn	4536
Saturn	-	4536
	Saab	5484
	Studebaker	30290
	Honda	14979
	Land	30755
	Nissan	14379

5.3 SUMMARY OF CONTRIBUTIONS

Given the results that were presented in the previous section, it is possible to conclude that our proposed attention networks are not only good price predictors on their own, but also that its representation captures additional information that is not present when using the original feature space. These facts are corroborated by some key observations from the experiments: (i) Every learning algorithm showed better performance using the car embedding than any combination of feature (structured, unstructured or both) in the original space; (ii)

The embedding CE performs better than STR+TXT, despite being learned with no additional feature. This suggests that the co-attention mechanism is indeed able to capture additional information from the interaction of both modalities and enrich their representation. (iii) A two-dimensional projection of the CE displays desirable properties of embeddings. Specifically, similarly-priced vehicles are close to each other in the space, and the projection shows a gradual change in price from one end of the spectrum to the other; (iv) The embeddings which encode categorical values also learn useful representations. The car manufacturer embedding, for instance, is able to retrieve similarly-priced manufacturers in terms of price when queried for the closest points to some manufacturer.

6 CONCLUSIONS AND FUTURE WORK

In this chapter, we present a brief overview of this study, highlighting its contributions, limitations and future research directions.

6.1 CONCLUSION

In this work, we propose a neural network that handles structured and unstructured data to perform a regression task: product price prediction. To handle the structured information, it uses a regular MLP network and for the unstructured data, it applies a co-attention mechanism for enhancing the embeddings of categorical variables and unstructured data. These two networks are combined to perform the regression and, as a side outcome, it produces a vector representation of product offer passed as input. One can assume, therefore, that our network is a feature generator mapping the raw features of a product offer to a reduced space.

We evaluate the architecture in a dataset that was collected from car classifieds websites, predicting the listing price from its attributes, description and title. We show that our model outperforms traditional regressors using raw features, providing further evidence that Deep Learning is suitable for learning on structured data. Furthermore, our learned embedding boosts the performance of all evaluated regressors by a sizable margin in comparison to using the original feature space. Lastly, we show that one of the intermediate embeddings associated with the car manufacturer categorical variable and the final representation learned by our model display interesting properties regarding the way they embed similarly-priced entries.

6.2 CONTRIBUTIONS AND LIMITATIONS

The main contributions presented in this work are:

1. An architecture for modeling problems that involve unstructured and structured data for predicting tasks;

2. A co-attention mechanisms for learning a co-dependent representation based on categorical variables and text, capable of enriching the representations in each modality by exploring interactions between them;

3. An experimental evaluation that reinforces the use of Deep Learning models to simplify the task for classical learning algorithms, while achieving an increased performance.

4. A new dataset based on real-world data containing structured and unstructured about car classifieds from multiple sources with different characteristics;

The main limitation of our approach is that it is hard to model the interactions between numerical variables and the text in the co-attention calculation. Some numerical variables, such as the mileage of a vehicle, can assume a large number of values and only rarely have repeated values. For this reason, most approaches remove or replace numerical tokens in the preprocessing step, so there are no embeddings associated with those numbers. Besides, even if those tokens are kept, it is unlikely that their representations exhibit the numerical properties that we expect. For example, the embedding of the token "10" added with the embedding of the token "20" might not be even remotely in the vicinity of the embedding of the token "30". This is because the syntactical statistics modeled by word embedding models do not capture mathematical notions, e.g. operations, order, etc.

6.3 FUTURE WORK

There area many clear paths to extend this work:

- Using numerical attributes in the co-attention computation. This might be done by quantizing the numerical tokens in the text;

- Evaluating the learned representations (CE) in other tasks other than price prediction, with and without domain adaptation;

- Employing subword and multi-word level features for representing tokens. This way, out-of-vocabulary tokens can still be represented by combining the representations of its character-level n-grams, and multi-word categories can be correctly matched in text, rather than being broken down into multiple tokens;

- Including other modalities, such as images;

- Expand the problem to other domains (houses, electronics, etc), more general domains (products) or multiple domains simultaneously;

- Explore the proposed architecture for solving other tasks, such as classification, retrieval, generating one modality from the other, or evaluating in a multi-task training setting;
- Visual analysis of the effect of the mutual contextualization of the co-attention mechanisms on the representations of categorical attributes and the text;
- More analysis on what makes similar products have significantly different prices;
- Leverage the abundance of unlabeled data which might contain structured and unstructured information in an unsupervised setting, such as generative pre-training, enhanced with co-attention.

REFERENCES

AIELLO, S.; ECKSTRAND, E.; FU, A.; LANDRY, M.; ABOYOUN, P. *Fast Scalable R with H2O*. [S.l.], 2015. Available at: <http://h2o.ai/resources>.

BAHDANAU, D.; CHO, K. H.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, p. 1–15, 2015.

BALLARD, D. H. Modular Learning in Neural Networks. *Aaai*, p. 279–284, 1987.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1798–1828, 2013. ISSN 01628828.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JANVIN, C. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, v. 3, p. 1137–1155, 2003. ISSN 15324435.

BITVAI, Z.; COHN, T. Non-linear text regression with a deep convolutional neural network. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, v. 2, p. 180–185, 2015.

BITVAI, Z.; COHN, T. Predicting peer-to-peer loan rates using Bayesian non-linear regression. *Proceedings of the National Conference on Artificial Intelligence*, v. 3, p. 2203–2209, 2015.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent Dirichlet allocation. *Journal of Machine Learning Research*, v. 3, n. 4-5, p. 993–1022, 2003. ISSN 15324435.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, oct 2001. ISSN 08856125.

BROMLEY, J.; BENTZ, J. W.; BOTTOU, L.; GUYON, I.; LECUN, Y.; MOORE, C.; SÄCKINGER, E.; SHAH, R. Signature Verification Using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 07, n. 04, p. 669–688, aug 1993. ISSN 0218-0014.

CHANDAR, S.; KHAPRA, M. M.; LAROCHELLE, H.; RAVINDRAN, B. Correlational neural networks. *Neural Computation*, v. 28, n. 2, p. 257–285, 2016. ISSN 1530888X.

CHANG, C. C.; LIN, C. J. LIBSVM: A Library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, v. 2, n. 3, p. 1–39, 2011. ISSN 21576904.

CHENG, J.; DONG, L.; LAPATA, M. Long short-term memory-networks for machine reading. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, p. 551–561, 2016.

CHO, K.; Van Merriënboer, B.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, p. 1724–1734, 2014.

CHOPRA, S.; HADSELL, R.; LECUN, Y. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.]: IEEE, 1997. v. 1, n. 1, p. 539–546. ISBN 0-7695-2372-2. ISSN 00143065.

CUI, Y.; CHEN, Z.; WEI, S.; WANG, S.; LIU, T.; HU, G. Attention-over-attention neural networks for reading comprehension. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, v. 1, p. 593–602, 2017.

DERELI, N.; SARACLAR, M. Convolutional Neural Networks for Financial Text Regression. p. 331–337, 2019.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. n. Mlm, 2018.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980. ISSN 03401200.

GAL, Y.; GHAHRAMANI, Z. A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, p. 1027–1035, 2016. ISSN 10495258.

GENTZKOW, M.; KELLY, B.; TADDY, M. Text as data. *Journal of Economic Literature*, v. 57, n. 3, p. 535–574, 2019. ISSN 00220515.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: *Journal of Machine Learning Research*. [S.l.: s.n.], 2011. v. 15, p. 315–323. ISSN 15324435.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. *Nature Methods*, v. 13, n. 1, p. 35–35, 2015. ISSN 1548-7091.

GUO, C.; BERKHAHN, F. Entity Embeddings of Categorical Variables. apr 2016.

HOCHREITER, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 06, n. 02, p. 107–116, apr 1991. ISSN 0218-4885.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. ISSN 08997667.

HOERL, A. E.; KENNARD, R. W. Ridge Regression: Applications to Nonorthogonal Problems. *Technometrics*, v. 12, n. 1, p. 69–82, 1970. ISSN 15372723.

HOFFER, E.; AILON, N. Deep metric learning using triplet network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9370, n. 1271, p. 84–92, 2015. ISSN 16113349.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, v. 1, p. 328–339, 2018.

HU, B.; SHI, C.; ZHAO, W. X.; YU, P. S. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, v. 1, p. 1531–1540, 2018.

HUANG, Z.; WANG, X.; HUANG, L.; HUANG, C.; WEI, Y.; LIU, W. CCNet: Criss-Cross Attention for Semantic Segmentation. 2018.

HUBER, P. J. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, v. 35, n. 1, p. 73–101, 1964. ISSN 0003-4851.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *32nd International Conference on Machine Learning, ICML 2015*. [S.l.]: International Machine Learning Society (IMLS), 2015. v. 1, p. 448–456. ISBN 9781510810587.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, v. 28, n. 1, p. 11–21, 1972. ISSN 00220418.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of tricks for efficient text classification. *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, v. 2, p. 427–431, 2017.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T. Y. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, v. 2017-Decem, n. Nips, p. 3147–3155, 2017. ISSN 10495258.

KIM, Y. Convolutional neural networks for sentence classification. In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. [S.l.]: Association for Computational Linguistics (ACL), 2014. p. 1746–1751. ISBN 9781937284961.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, p. 1–15, 2015.

KNYAZEV, B.; SHVETSOV, R.; EFREMOVA, N.; KUHARENKO, A. Convolutional neural networks pretrained on large face recognition datasets for emotion classification from video. p. 2–5, 2017.

LIU, M.; YIN, H. Cross Attention Network for Semantic Segmentation. *British Machine Vision Conference 2018, BMVC 2018*, jul 2019. ISSN 18728286.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

LU, J.; YANG, J.; BATRA, D.; PARIKH, D. Hierarchical Question-Image Co-Attention for Visual Question Answering. *Bollettino chimico farmaceutico*, v. 90, n. 9, p. 343–348, may 2016.

LUONG, T.; PHAM, H.; MANNING, C. D. Effective Approaches to Attention-based Neural Machine Translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2015. p. 1412–1421. ISBN 9781941643327.

MENG, L.; LI, Y. Multi-Modal Similarity Metric Learning for Answer Selection. p. 1–5, 2016.

MIKOLOV, T. Efficient Estimation of Word Representations in Vector Space. *IJCAI International Joint Conference on Artificial Intelligence*, v. 2015-Janua, p. 4069–4076, 2013. ISSN 10450823.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 1389–1399, oct 2013.

NL, L. V. U.; HINTON, G. Visualizing Data using t-SNE Laurens van der Maaten. *Journal of Machine Learning Research*, v. 1, p. 1–48, 2008. ISSN 1532-4435.

NOGUEIRA, R. F.; De Alencar Lotufo, R.; Campos MacHado, R. Fingerprint Liveness Detection Using Convolutional Neural Networks. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 11, n. 6, p. 1206–1213, 2016. ISSN 15566013.

PAETZOLD, G.; SPECIA, L. Feature-Enriched Character-Level Convolutions for Text Regression. v. 2, p. 575–581, 2018.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, p. 1532–1543, 2014. ISSN 10495258.

PETERS, M.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep Contextualized Word Representations. p. 2227–2237, 2018.

PHAM, N. Q.; KRUSZEWSKI, G.; BOLEDA, G. Convolutional neural network language models. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, p. 1153–1162, 2016.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2020. Available at: <https://www.R-project.org>.

RADFORD, A.; JOZEFOWICZ, R.; SUTSKEVER, I. Learning to Generate Reviews and Discovering Sentiment. 2016.

RADFORD, A.; SALIMANS, T. Improving Language Understanding by Generative Pre-Training. *OpenAI*, p. 1–12, 2018.

RAO, J.; LIU, L.; TAY, Y.; YANG, W.; SHI, P.; LIN, J. Bridging the Gap between Relevance Matching and Semantic Matching for Short Text Similarity Modeling. p. 5373–5384, 2019.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, 1986. ISSN 00280836.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, p. 567, 1987.

SANTOS, C. dos; TAN, M.; XIANG, B.; ZHOU, B. Attentive Pooling Networks. n. Cv, 2016.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, v. 07-12-June-2015, p. 815–823, 2015. ISSN 10636919.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, p. 1929–1958, 2014. ISSN 15337928.

STEINER, B.; DEVITO, Z.; CHINTALA, S.; GROSS, S.; PASZKE, A.; MASSA, F.; LERER, A.; CHANAN, G.; LIN, Z.; YANG, E.; DESMAISON, A.; TEJANI, A.; KOPF, A.; BRADBURY, J.; ANTIGA, L.; RAISON, M.; GIMELSHEIN, N.; CHILAMKURTHY, S.; KILLEEN, T.; FANG, L.; BAI, J. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeuroIPS*, n. NeurIPS, 2019.

SUBRAMANIAN, S.; BALDWIN, T.; COHN, T. Content-based popularity prediction of online petitions using a deep regression model. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, v. 2, p. 182–188, 2018.

SUBRAMANIAN, S.; TRISCHLER, A.; BENGIO, Y.; PAL, C. J. Learning general purpose distributed sentence representations via large scale multitask learning. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, n. 2016, p. 1–16, 2018.

SUTSKEVER, I.; MARTENS, J.; DAHL, G.; HINTON, G. On the importance of initialization and momentum in deep learning. In: *30th International Conference on Machine Learning, ICML 2013*. [S.l.: s.n.], 2013.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, v. 4, n. January, p. 3104–3112, 2014. ISSN 10495258.

TAY, Y.; PHAN, M. C.; TUAN, L. A.; HUI, S. C. SKIPFLOW: Incorporating neural coherence features for end-to-end automatic text scoring. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, p. 5948–5955, 2018.

TIBSHIRANI, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B*, v. 58, n. 1, p. 267–288, 2016.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [S.l.]: Neural information processing systems foundation, 2017. v. 2017-Decem, p. 5999–6009. ISSN 10495258.

WU, L.; FISCH, A.; CHOPRA, S.; ADAMS, K.; BORDES, A.; WESTON, J. StarSpace: Embed all the things! *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, p. 5569–5577, 2018.

XIONG, C.; ZHONG, V.; SOCHER, R. Dynamic coattention networks for question answering. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, p. 1–14, 2019.

YIN, W.; SCHÜTZE, H.; XIANG, B.; ZHOU, B. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, v. 4, p. 259–272, dec 2016. ISSN 2307-387X.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, M. Y. A Structured Self-Attentive Sentence Embedding. *Iclr*, p. 2, 2017.

ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, v. 67, n. 2, p. 301–320, apr 2005. ISSN 1369-7412.