University Institute of Lisbon

Department of Information Science and Technology

Attribute-Value Inference using Deep Neural Networks

Kevin Almeida Ramos

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Science and Business Management

Supervisor

Ricardo Daniel Santos Faro Marques Ribeiro, Assistant Professor, Ph.D.

ISCTE-IUL

Supervisor

Sancho Moura Oliveira, Assistant Professor, Ph.D.

ISCTE-IUL

October, 2019

"We are prisoners of the present, in perpetual transition from an inaccessible past to an unknowable future."

- Neil deGrasse Tyson on Twitter, 2019

Abstract

The population's consumption patterns have changed over the last few years and e-commerce has been one of the main drivers. The consumer became very demanding and very knowledgeable about the product and the websites were adapting, providing more information and improving the filtering system by adding detailed descriptions of the products and their characteristics. Extracting different characteristics from thousands of products is a task with a very high cost. In this work, we created three datasets that were later used by our model with three layers, CNN-BiLSTM-CRF, to infer values of attributes of previously unknown products through the description of products. It inferred with 64% of macro average of f1-score, not being related to the state of the art due to the different context of the tests.

Keywords: Named Entity Recognition, Information Extraction, Neural Sequence Labeling

Resumo

Os padrões de consumo da população alteraram-se nos últimos anos e o e-commerce foi um dos grandes responsáveis. O consumidor tornou-se muito exigente e bastante conhecedor do produto e os websites foram-se adaptando, disponibilizando mais informação e melhorando o sistema de filtragem, adicionando descrições detalhadas dos produtos e as suas características. Extrair diferentes características de milhares de produtos é uma tarefa com um custo bastante elevado. Neste trabalho, criamos três conjuntos de dados que posteriormente foram usados pelo nosso modelo com três camadas, CNN-BiLSTM-CRF, para inferir valores de atributos de produtos anteriormente desconhecidos através da descrição dos produtos. Inferiu com 64% de macro média de f1-score, não sendo relacionável com o estado de arte devido ao contexto dos testes serem distinto.

Palavras-chave: Reconhecimento do nome da entidade, Extração de Informação, Etiquetagem Sequencial Neuronal

Acknowledgements

To my supervisors, Professor Ricardo Ribeiro and Professor Sancho Oliveira I would like to thank them for sharing their visions, knowledge and mentoring.

To the Instituto de Telecomunicações for the supply of all the necessary material.

To my friends and colleagues who helped me on this long journey, thank you for the good times.

To the family, and especially to my parents for all their tireless support, always supporting me to continue my academic education and to follow my dreams.

Contents

List of Tables

List of Figures

Acronyms

AVP Attribute-Value Pairs. 1–3, 24, 25

BiLSTM Bidirectional LSTM. xii, 2, 17, 26, 27, 36, 38–41, 53

CNN Convolutional Neural Network. xv, 2, 18, 19, 26, 27, 36, 41, 53

CRF Conditional Random Fields. xii, xv, 2, 7, 11, 20, 21, 25–27, 36, 39, 41, 53

DSR Design Science Research. 4

KB Knowledge Base. 25

LSTM Long Short-Term Memory. 16, 17, 27, 38

NB Naive Bayes. 24, 25

NER Named Entity Recognition. 11, 23, 26

NLP Natural Language Processing. 2, 9, 14, 21, 36

POS Part-of-Speech. 10

RNN Recurrent Neural Network. 15, 16

VG Vanishing Gradient. 16

Chapter 1

Introduction

Attribute-Value Extraction is a research area of high interest within the information retrieval and text mining community. Deconstructing the product description in Attribute-Value Pairs (**AVP**) is the main goal of this work.

Marketplaces like Amazon[1], eBay[2], AliExpress[3] as well as niche websites like Tiffany & Co.[4] for jewelry products or FTD[5] that deals with flowers, have gained popularity. The flow of information, knowledge and rapid delivery has changed consumer consumption patterns.

According to a (Statista, 2019) study, between 2017 and 2019, there was a 13.16% growth in the number of users who prefer to shop online instead of going to the physical store, and between 2019 and 2023 they forecast a 17.60% increase (Figure 1.1).

Large retailers as well as niche stores index a large number of products in their respective scales, which allows for greater consumer choice but raises concerns due to the need to specify and create filtering tools to make it easier for users to search.

According to Baymard's Product Listings & Filtering study, "sites with mediocre product list usability saw abandonment rates of 67-90%, (...)"(Baymard, 2019).

[1] https://www.amazon.com/
[2] https://www.ebay.com
[3] https://www.aliexpress.com
[4] https://www.tiffany.com/
[5] https://www.ftd.com/

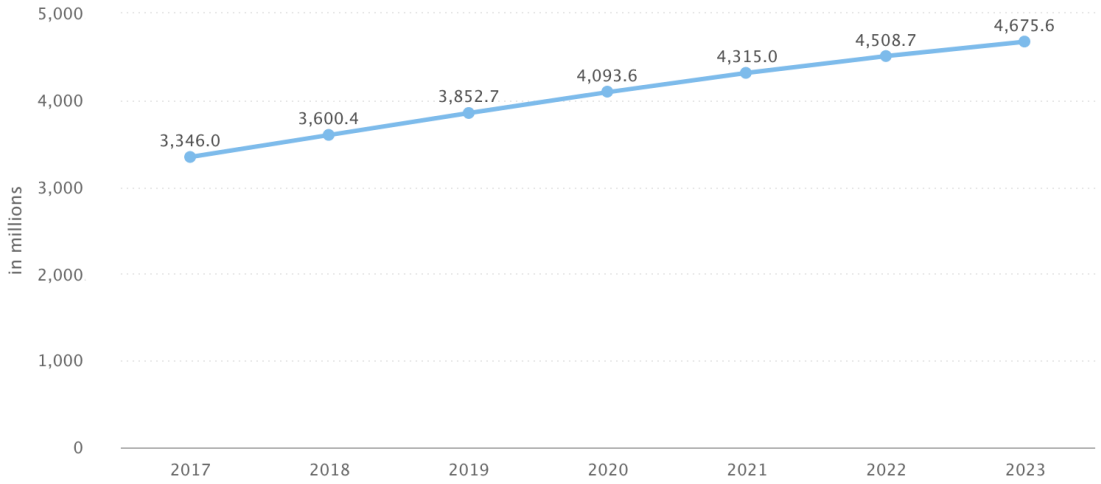

FIGURE 1.1: Evolution and expectation of the number of e-Commerce users (Statista, 2019).

Brands generally provide the title, description and images of products. Due to the amount of information in natural language provided by descriptions, the need for a mechanism capable of extracting the pairs of attributes and values has arisen.

1.1 Thesis Proposal

In the context of this dissertation project, the aim is to develop a model capable of inferencing **AVP** automatically from e-Commerce products description.

Developing a model for this purpose is particularly interesting because of the huge amount of data that e-Commerce companies have about consumer consumption patterns. Data is today a key piece in the strategies of these websites.

Natural Language Processing (**NLP**) plays a crucial role in text interpretation due to the text being in an unstructured format.

Our proposed approach uses Convolutional Neural Network (**CNN**) and neural sequence labeling algorithms, in this case, Bidirectional LSTM (**BiLSTM**) and Conditional Random Fields (**CRF**), checking the generalization between datasets of different categories.

It is therefore important to find out how precisely the tagging of **AVP** can be classified in this context, as well as to understand to what extent the current limitations of the datasets may or may not make its applicability in a real context.

1.2 Research Questions and Objectives

This dissertation project aims to create a model capable of detection and tagging **AVP** from e-Commerce product description, obtaining results through training a neural networks model to capture the correspondence between the attribute and values.

The validation of the proposed model should be done through the performance metrics used to achieve the maximum F1-Score.

As so, the main objectives are the following:

- Improve a model of attribute value pairs extraction from product description based on small dataset.
- Test the generalization capability of a sequential labeling model from one dataset to another.

With the objectives established above, the major questions to be answered in this present work are:

1. Determine the most appropriated tagged attributes.
2. Determine the maximum F1-score.
3. Determine the maximum F1-score of generalization.

1.3 Methodological Approach

Scientific research requires rigor, quality of the proposal developed and be subject to debate and verification with the community.

There are quality standards that are relevant and applicable in certain areas of research, in this context the methodological proposal Design Science Research (**DSR**) was relevant (Kuechler and Petter, 2004).

"Research by (Kuechler and Petter, 2004) has provided evidence that..." there are two crucial activities to understand the problem and try to innovate information systems: "(1) the creation of new knowledge through design of novel or innovative artifacts (things or processes) and (2) the analysis of the artifact's use and/or performance with reflection and abstraction (Kuechler and Petter, 2004, p. 13)".

In Figure 1.2, the cycle of the **DSR** process are represented, highlighting the five steps of the process: (1) Awareness of problem, (2) Suggestion, (3) Development, (4) Evaluation and (5) Conclusion. Throughout the dissertation, we will establish the parallelism between the work developed and the correspondence of the above mentioned processes.

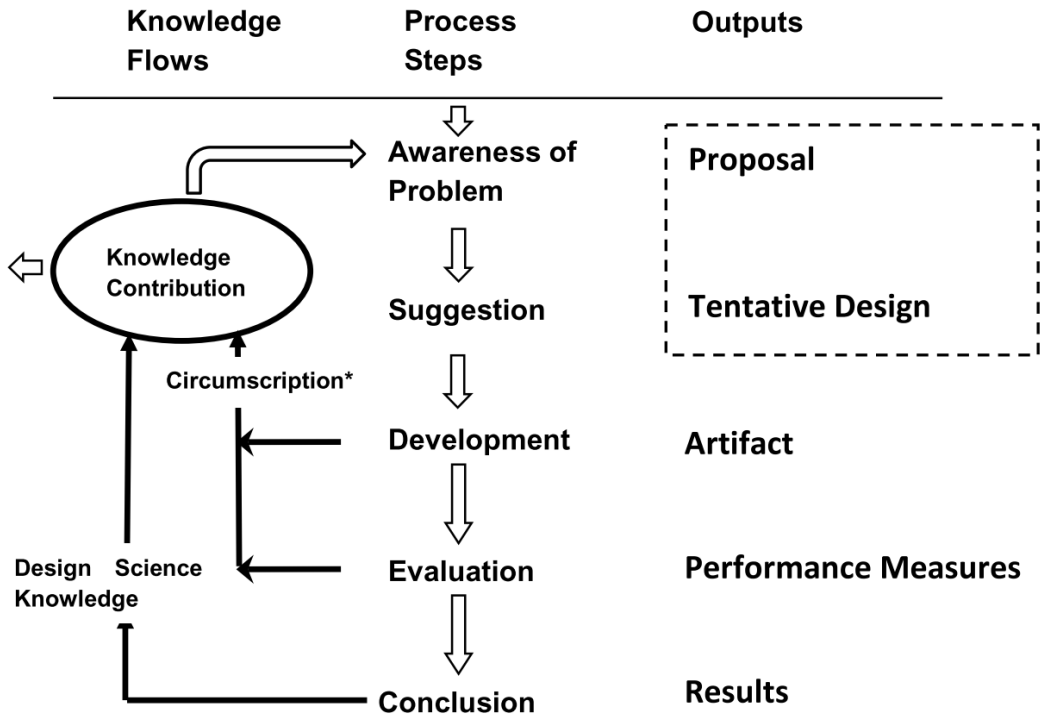

FIGURE 1.2: Design Science Research Process Model (Kuechler and Petter, 2004)

Based on Figure 1.3, the expected output knowledge contribution of this project is adaptation. (i.e, "non-trivial or innovative adaptation of known knowledge/solutions for new problems (Kuechler and Petter, 2004, p.13)").

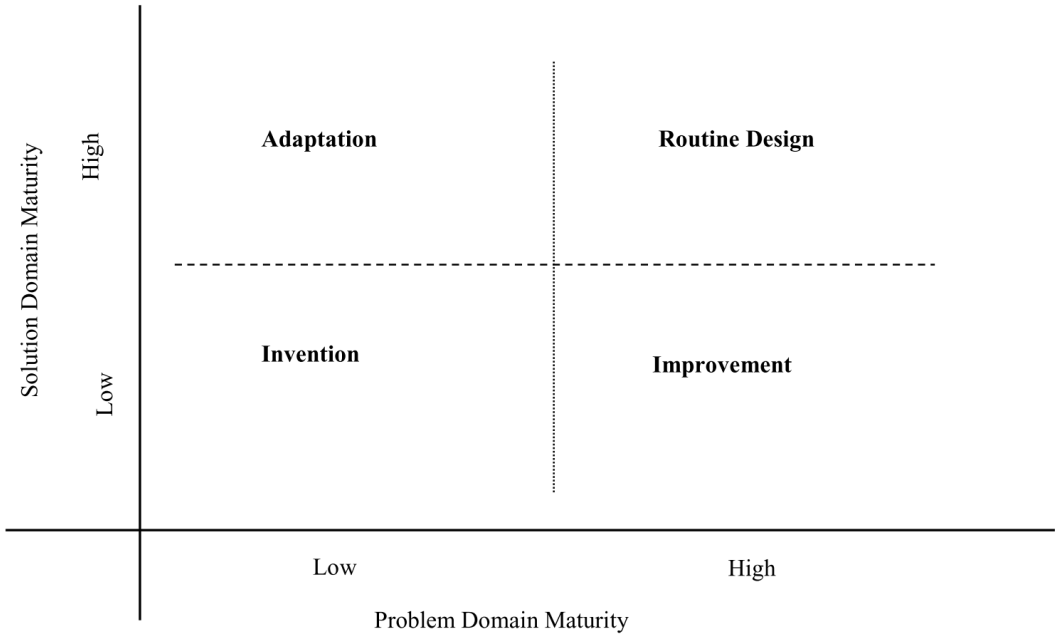

FIGURE 1.3: DSR Knowledge Contribution Framework (adapted from Gregor and Hevner, 2013)

1.4 Document Structure

In order to present an overview of the structure of the dissertation project and to meet the best practices against the selected research methodology (Section 1.3), a summary table was prepared where it is possible to observe the division as well as its description.

Chapter	**Design Science Research Phases**
1- Introduction	Problem Identification
2- Fundamental Concepts	Identify and explain the fundamental concepts
3- Related Work	Identify related work
4- Development Work	Identification and describe the datasets used. Proposed Solution. Development of the proposed solution
5- Experimental Settings and Evaluation	Explain test metrics. Extraction of results. Evaluation
6- Conclusion and Future Work	Summarizing the main findings of this work. Identify which are the remaining future challenges.

TABLE 1.1: Mapping of chapters against process steps associated with the DSR methodology

Based on the problem previously identified in Chapter 1, in Chapter 2 we present the fundamental concepts in order to understand the technology behind the attribute value pair extraction and neural sequence labeling models.

Mathematical algorithms as well as state-of-the-art approaches are provided in Chapter 3.

Chapter 4 shows the work done and the technology used at each step.

Chapter 5 then presents the experimental settings and evaluation: the datasets used, evaluation metrics, experimental settings and results.

Finally, Chapter 6 concludes this document by summarizing the main conclusions of this work, highlighting possible directions for future research.

Chapter 2

Fundamental Concepts

As mentioned in chapter 1, extracting the attribute-value pairs manually is too costly and time-consuming. In order to make the task more efficient and effective, it is necessary to understand the concepts and architecture behind the extraction of the attribute-value pairs.

This chapter provides an overview of the product concept, attribute and attribute value. It also provides an introduction of natural language processing levels: tokenization, part-of-speech and named entities. An introduction about how the word embeddings works and the different existing types as well as neural network models and Conditional Random Fields (**CRF**).

2.1 Product, Attribute and Value Concept

Product is something that is made to be sold by retailers, it can be tangible or intangible.

Attribute is a characteristic of a product which make it distinct from other products. Attribute can also be representative of all products belonging to a list.

Value of the attribute is associated to an object and the values that an attribute can have are determined by the type of the represented attribute.

Values of product attributes can assume three types: string, boolean and numbers.

The development of e-Commerce websites has evolved over time and advanced product search through the search field or attribute filtering is already part of good development practice.

Nowadays, most retailers understand what business efficiency is about, they adapt and use data to their advantage, improving their recommendation systems, demand forecasting, assortment comparison and optimization.

Figure 2.1 shows the Charme 24[1] filtering system which presents some product characteristics but in this specific products type denounces limitations in the filtering implementation, such as product color, stones and product finish.

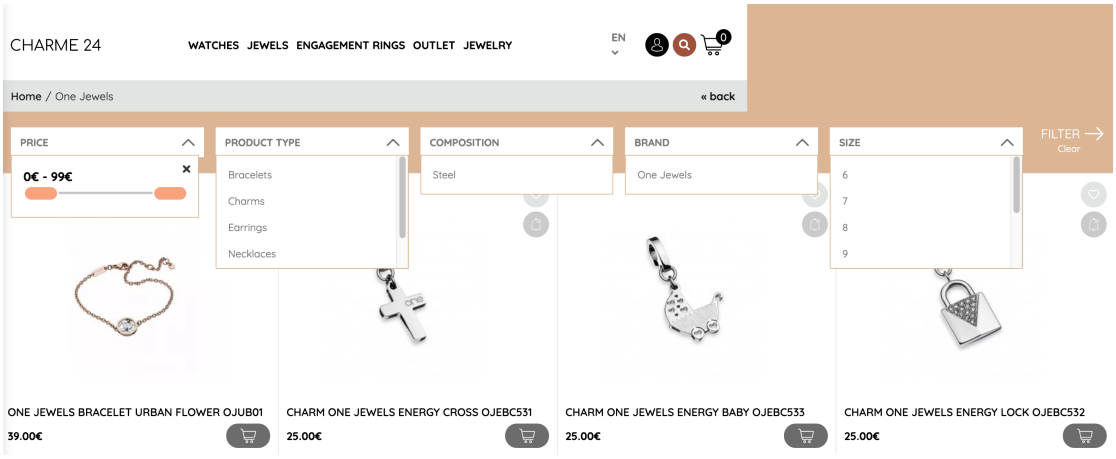

FIGURE 2.1: Example of a filtering system of the Charme 24 - from https://www.charme24.com/en.

Some features are more relevant in customer decision making than others, however, a good filtering system is important as feature on an e-Commerce website.

In Chapter 4, the structure is explained in detail, as well as the characteristics and values that can be obtained from the product description present in the dataset used in the development of this project.

2.2 Natural Language Processing Levels

We divide this section in three levels: tokenization(Section 2.2.1), part-of-speech(Section 2.2.2) and named entities(Section 2.2.3).

[1]https://www.charme24.com/en

Figure 2.2 illustrates the pipeline architecture for a typical Information Extraction system.

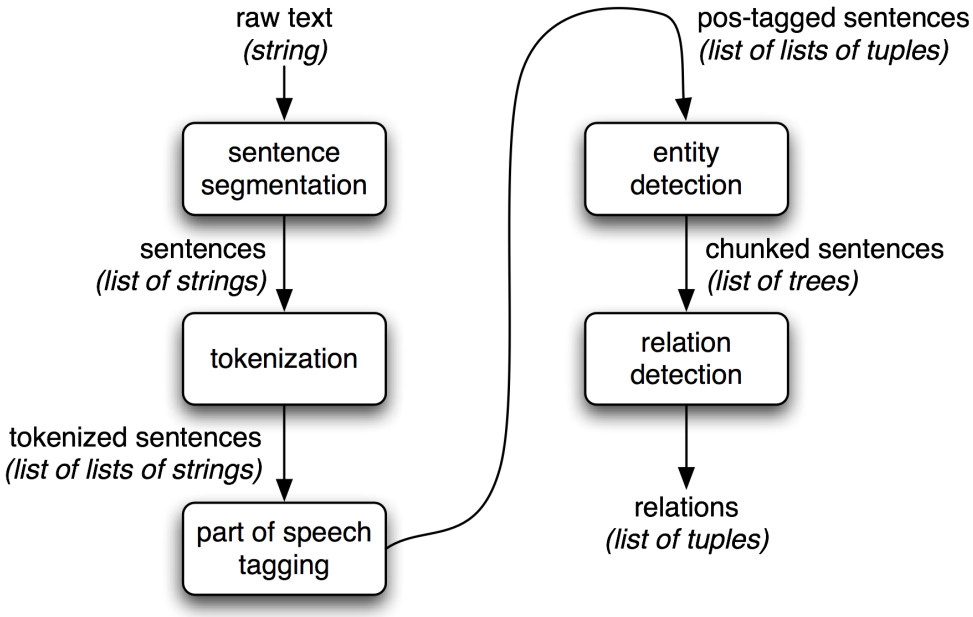

FIGURE 2.2: Pipeline Architecture for an Information Extraction System(Bird et al., 2009).

2.2.1 Tokenization

Tokenization is responsible for processing the text and transforming it into tokens. This process can also be known as lexer or tokenizer and is one of the first steps in **NLP**.

$$X = Casio\ Edifice\ Retrograde\ Chronograph\ Watch\ Men\ EFV\text{-}530GL\text{-}5AVUEF$$

where X is a product description and $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ be a particular tokenization X_t of X.

Below is a representation of the tokens from product description after using whitespace tokenization[2].

$$X_t = (x_1, x_2, x_3, x_4, ...) = (Casio, Edifice, Retrograde, Cronograph, Watch, ...)$$

[2]The whitespace tokenizer breaks text into tokens whenever it encounters a whitespace character.

2.2.2 Part-of-Speech

- **Part-of-Speech (POS) Tagging** is the process of assigning a token in a corpus the corresponding part of a speech tag, based on its context and definition.

There are different techniques of **POS** tagging, the choice depends on the decision behind the algorithm to use for the problem in question.

Probabilistic methods are commonly used, where n-grams are specially important because "picks the tag that is most likely in the given context" (Bird et al., 2009, p.204) using the previous ones to calculate.

In Figure 2.3, an example of how tagger context works using n-gram tagging.

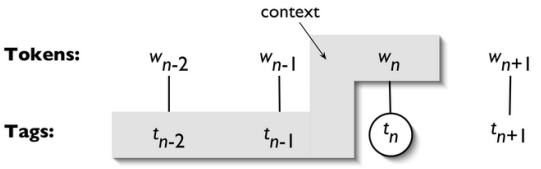

FIGURE 2.3: Example of tagger context works using n-gram (Bird et al., 2009).

For this example below, we will be using the tagset of the *The Penn Treebank* of (Marcus et al., 1993) to show the importance of a correct meaning given to a word to determine its classification as a **POS**.

Tag	Description
DT	Determiner
IN	Preposition or subordinating conjunction
JJ	Adjective
NN	Noun, singular or mass
NNP	Proper noun, singular
NNPS	Proper noun, plural
VBD	Verb, past tense
VBZ	Verb, 3rd person singular present

TABLE 2.1: Example of POS Tags from Penn Treebank Project.

Sentence 1: "The/DT earth/NN is/VBZ round/JJ"

Sentence 2: "Portugal/NNP won/VBD the/DT first/JJ/ round/NN of/IN UEFA/NNP Nations/NNPS League/NNP"

The word *round* is the same but the meaning is different. On the first one is a Adjective, in the second is a Noun.

2.2.3 Named Entities

Named entities is the task responsible for identifying all the noun phrases that correspond to a certain specific type (people, places, organization, dates and so on) that are mentioned in the string of text passed as input.

- **Named Entity Recognition (NER)** is the process of identify all named entities. **NER** is a technique applied in many areas, such as question-answering, summarization, and machine translation. It is a widely used due to its effective approach that allows a reduction in search time, the algorithm directs its search according to the entities it finds (Nadeau and Sekine, 2007).

The appearance of characteristics previously unknown to the system is a hindrance to their scalability.

Supervised, semi-supervised and unsupervised machine learning are the methods used to recognize and tag named entities, standing out for the huge adoption of supervised and semi-supervised methods where Hidden Markov Model and Conditional Random Fields (**CRF**) have had a great performance in this type of tasks and recently for neural networks models that are the state-of-the-art (Zheng et al., 2018; Yadav et al., 2018; Dirie, 2017).

The adoption of the method involves the type of problem to be solved and based on the available dataset, however, the BIO Tagging Scheme is a chunk tag set method widely used in this type of tasks.

Tag	Description
B	Beginning of a chunk
I	Inside of a chunk
O	Outside of a chunk

TABLE 2.2: BIO Chunk Tag scheme.

In the example below we show how the system works, especially the *UEFA Nations League* where you can see the distinction between the beginning of the name of the organization *UEFA/B-ORG* and the remaining words that complete the name *Nations/I-ORG League/I-ORG*.

Sentence: "Portugal/B-GPE won/O the/O first/O round/O of/O UEFA/B-ORG Nations/I-ORG League/I-ORG"

In the previous example two named entities were recognized. *Portugal* was labeled as a country (GPE) and the *UEFA Nations League* was labeled as an organization (ORG).

Due to the fact that e-Commerce has specific named entities, we had to create custom named entities. In Section 4.1.3, the custom named entities are explained in detail, as well as the tags and values that can be obtained from the product description present in the datasets used in the development of this project.

2.3 Word Embeddings

"One-hot" encoding was the first attempt to represent text in vectors.

Each word is represented by a vector where its dimension is equal to the size of the vocabulary. A vector is composed of zeros and a single one at the position of the represented word.

The vocabulary ['cat', 'mat', 'on', 'sat', 'the'] represented in Figure 2.4, illustrates the previous explanation.

In terms of scale and performance, this model proved to be quite weak due to the excessive amount of zeros and its inability to measure the similarity relationship between words.

Although there are slight changes in its representations, the structure that remains today is called word embeddings.

Word embeddings are a set of embeddings with dense vector representations that contain floating point values within each vector.

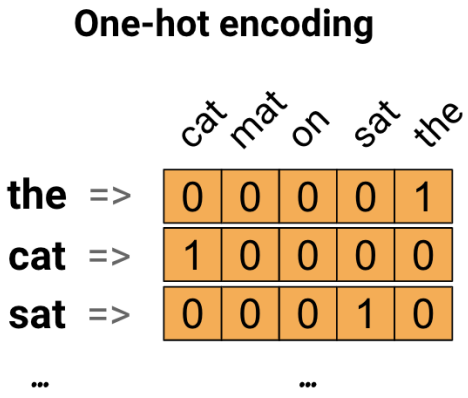

FIGURE 2.4: Example of a "one-hot" vector structure, from https://www.tensorflow.org/tutorials/text/word_embeddings.

The encoding in the vector is done in a way to represent the similarity relations between the words. The floating points are the weights of these relations and are automatically learned through the dataset used.

Their advantage comes from the capability of capturing the similarity of meaning of certain words, therefore the vector representation is close to capturing the relationship between them.

As illustrated in Figure 2.5, an embedding is represented by floating-point values.

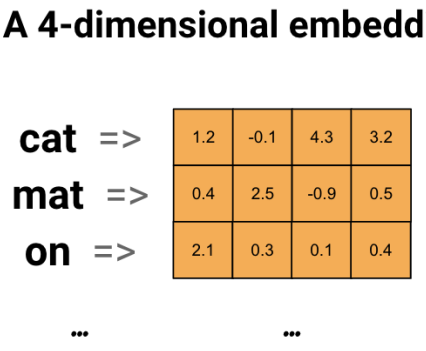

FIGURE 2.5: Example of a word embedding structure with the floating-point values represented, from https://www.tensorflow.org/tutorials/text/word_embeddings.

Word embeddings techniques have evolved, and in a generalized way can be represented as two approaches: classics or contextual. (Camacho-Collados and Pilehvar, 2018)

Classic Word Embedding

The classic techniques are known to be static word embeddings because a word only has a single representation no matter the context in which it occurred.

Word representation in vector space also known as Word2Vec continues to be one of the most widely used approaches due to fact that can be represented in two model architectures: CBOW created by (Mikolov et al., 2013a) and Skip-gram created by (Mikolov et al., 2013b).

Figure 2.6 shows the two approaches of Word2Vec.

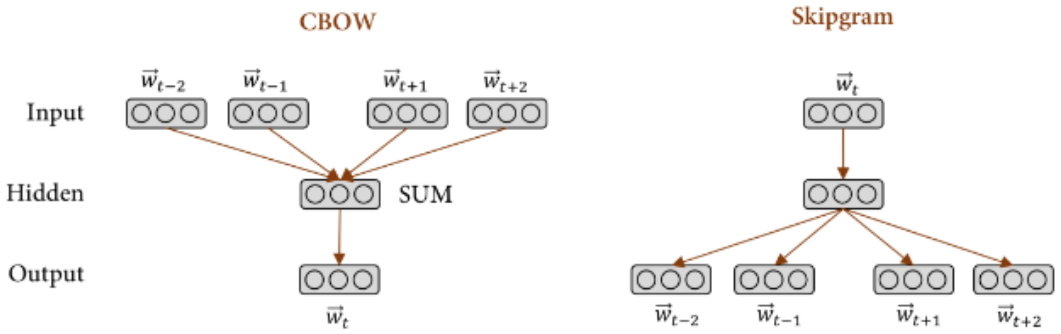

FIGURE 2.6: CBOW and Skip-gram models of Word2Vec. Adapted from (Camacho-Collados and Pilehvar, 2018).

FastText created by (Joulin et al., 2016) or Glove by (Pennington et al., 2014) are also some of the classic techniques that fails to capture the polysemy[3]. Typically, they perform a loop up, mapping a word to a vector.

Contextual Word Embedding

Recent techniques use language models to calculate the probability of the next word in a sequence of words using the context as a reference, thus creating contextualized word embeddings where it is possible to capture the semantics of words in different contexts, solving the problem of polysemy.

BERT presented by (Devlin et al., 2018) is the state of the art in **NLP** tasks and the fast fine tuning is its the major point. Figure 2.7, represents the architecture.

[3]Words with several different meanings

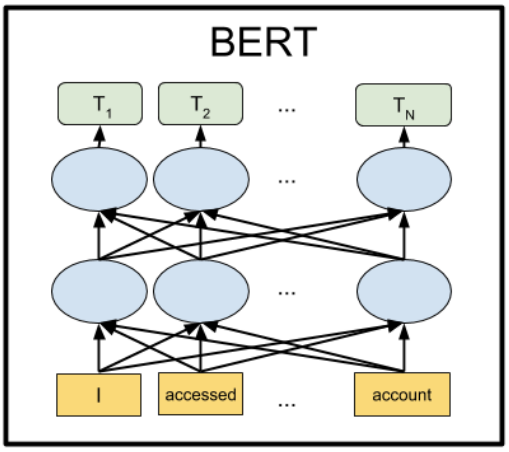

FIGURE 2.7: BERT model architecture. Adapted from https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

ELMO created by (Peters et al., 2018) or Flair Embeddings by (Akbik et al., 2018) has similarly to BERT the downside of being necessary a lot of computational power to perform this task.

2.4 Neural Networks

The basic architecture of a neural network is the feed-forward network, composed of several layers of neurons, where the leftmost layer in the network is called the input layer, the rightmost is the output layer and in the middle, we have the hidden layers. The design of the input and output layers in a network is often straightforward, which means that the output value of each neuron will be the input value of the next neuron, forming a single fully connected neural network (Figure 2.8).

The **RNN** are simply loops of feed-forward neural networks. In Figure 2.9, it is possible to observe a standard **RNN**. By decomposing a **RNN**, a chunk of the neural network is visible in the middle state, where x_t is the input, tanh is the activation function that defines the output given an input and h_t is the output value. A loop allows information to be passed from one step of the network to the next. In other words, a **RNN** is like a multiple copies of the same network, each passing a message to a successor.

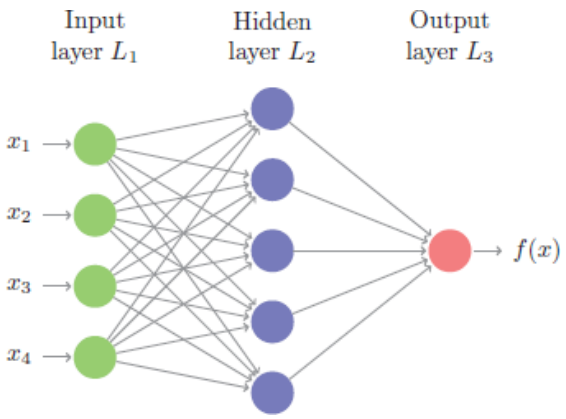

FIGURE 2.8: Simple feed-forward neural network, adapted from http://uc-r.github.io/feedforward_DNN

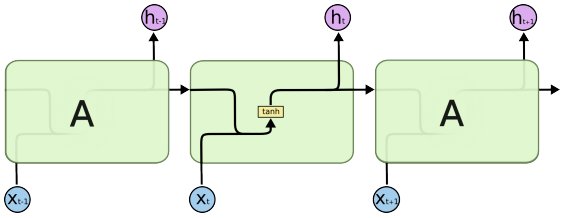

FIGURE 2.9: Standard Recurrent Neural Network single tanh layer, adapted from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Considering the architecture, it is evidenced that **RNN** are related to sequences and lists and hence the problem of Vanishing Gradient (**VG**).

The **VG** problem is related to the update of weights over backpropagation through time when modelling a long sequence of words, it is easily corrupted by multiplying small gradients over the sequence to the initial state. To overcome, this problem (Hochreiter and Schmidhuber, 1997) created a **RNN** variant called Long Short-Term Memory (**LSTM**).

RNN are being overlooked by **LSTM** due to the high performance of their hidden state layers. It increases the complexity of the model but allows a more effective solution.

Figure 2.10 presents the **LSTM** architecture and how its cells works. One determinant feature is the memory cell C_t and have the ability to influence the storing or overwriting memories. The formula is defined in Equation 2.1.

$$C_t = f_t * C_{t-1} + i_t * \check{C}_t \tag{2.1}$$

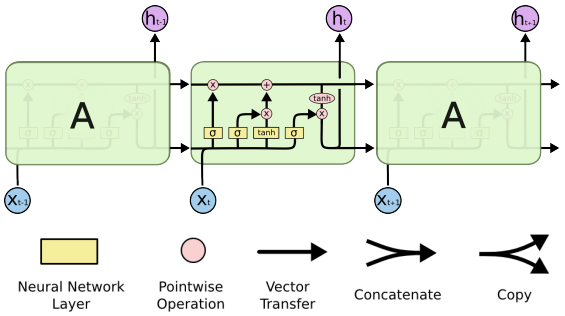

FIGURE 2.10: Long Short-Term Memory architecture, adapted from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Each cell is composed by three important gates: forget f_t, input i_t and output gate o_t. Those gates have a different role but they all block or pass information based on its strength and importance, filtered by their own sets of weights.

Forget gate layer has the responsibility to choose what information retain based on previous output layer h_{t-1} and input of the current cell x_t (Equation 2.2).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.2}$$

The next step is the input gate layer where learns new inputs that are worth using and determines how much of the input to let into the cell state (Equation 2.3).

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \check{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{2.3}$$

And the last gate are output layer where is decided what output value goes out and stored in the memory cell (Equation 2.4).

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{2.4}$$

Recent approaches are based on Bidirectional LSTM (**BiLSTM**). They are similar to **LSTM**s, but have advantage of accessing information in both directions. (i.e. by representing future steps, you can understand the context and eliminate ambiguity)

Also, we have the Convolutional Neural Network (**CNN**), which is a variation of the neural networks where the greats contributions are in the fields of computer vision and audio simply because convolution and pooling functions are used as activation functions (Figure 2.11).

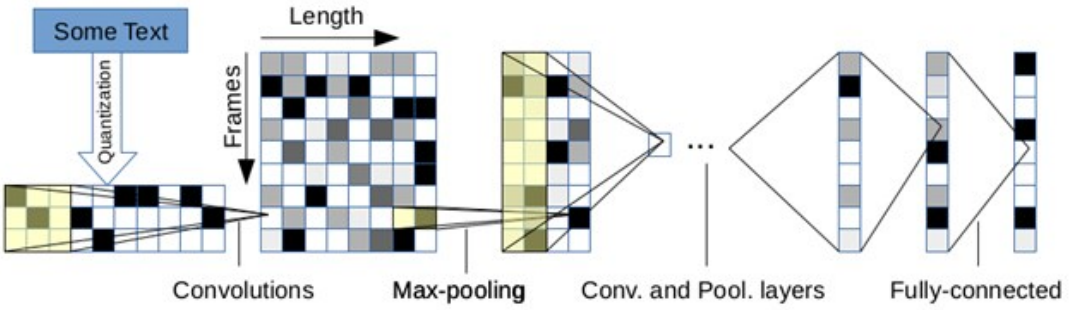

FIGURE 2.11: Example of **CNN** architecture (Zhang and LeCun, 2015).

Sparse interactions, parameter sharing, and equivariant representations are important characteristics of convolution.

The interactions between the input and output units are quite different from traditional neural networks.

The convolution layer aims to reduce the complexity of the data entered by trying to find relevant characteristics, reducing the number of parameters and memory used allowing an increase in efficiency in the model. By limiting the number of possible connections to the output, we are limiting the number of possible parameters and indirectly the number of runtimes.

The size of the weights, also known as filter or kernel, is smaller than the input data purposely to be applied multiple times at different input points. Thus, it can scroll efficiently through the input data from left to right, top to bottom (Figure 2.12).

Equation 2.5 demonstrated the formula behind discrete convolution.

$$(f * g)(i) = \sum_{j=1}^{m} g(j) \cdot f(i - j) \tag{2.5}$$

Where g is the input and f is the kernel, also this formula can only be defined if we assume that g and f are defined in the integer i.

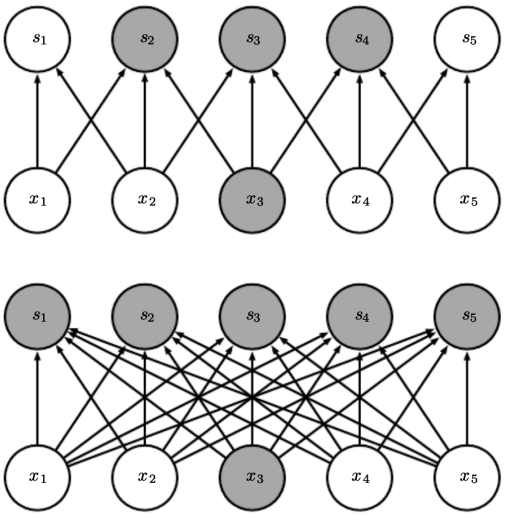

FIGURE 2.12: Example of sparse connectivity, convolution layer represented on top with three width kernel, in the bottom is represented a traditional matrix multiplication, where are not sparse (Goodfellow et al., 2016).

If we use convolutions with more than one axis in the same time reference, need to use a two-dimensional kernel as represented at Equation 2.6.

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \tag{2.6}$$

or Equation 2.7 because convolution are commutative.

$$(K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \tag{2.7}$$

Where I is a two-dimensional image input and K is a two-dimensional kernel.

As exhibited in Figure 2.13, pooling is very relevant in the composition of **CNN** since it allows us to give a fixed output value as well as the reduction of the dimensionality saving only the information considered useful and that stands out. It also has overfitting supervision as a characteristic and can assume the min polling layer or average polling layer, although the max polling layer is the most adopted.

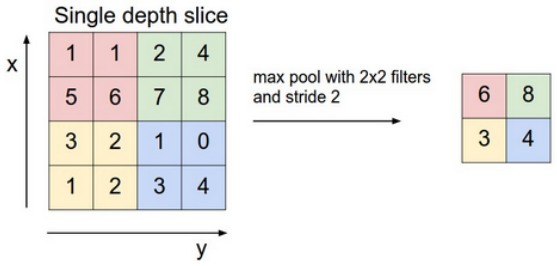

FIGURE 2.13: Example of max pooling layer effect. Adapted from http://cs231n.github.io/convolutional-networks/#pool.

2.5 Conditional Random Fields

Conditional Random Fields (**CRF**) originally proposed by (Lafferty et al., 2001) belongs to the group of discriminative classifiers, and they model the decision boundary between the different classes.

The discriminative models assume the following Equation 2.8, where Y and X are given directly from the training set.

$$P(Y|X) \tag{2.8}$$

The model has learned the decision boundary that separates the data points by learning the conditional probability distribution. However, it needs to be efficient to predict a sequence computation like Equation 2.9.

$$\hat{y} = argmax_y P(Y|X) \tag{2.9}$$

Like linear regression, the **CRF** also uses feature function to represent characteristics in data sequences.

The feature function is represented by Equation 2.10.

$$F(\bar{x}, \bar{y}) = \sum_i f(y_{i-1}, y_i, \bar{x}, i) \tag{2.10}$$

Where the $\bar{f}$ function analyzes the entire $\bar{x}$ sequence for the corresponding tags, $\bar{i}$ is the current position where it is in the phrase. $\bar{y}$ $\bar{y_{i-1}}$ represents the

previous positions in the tag sequence and $\bar{y}_i$ corresponds to the current position.

Areas such as part-of-speech tag or named entity recognition are very proliferating due to the need to predict the current word tag and its dependency on neighboring words and tags.

The model can assume several different graphs, our focus is on linear-chain **CRF** due to its structure and capability in **NLP** tasks.

Below is represented a linear-chain structure in the Figure 2.14 and is formulated using Equation 2.11.

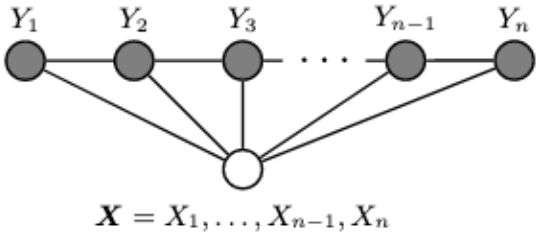

FIGURE 2.14: Linear chain-structured **CRF**s (Wallach, 2004).

$$P(\bar{y}|\bar{x};\bar{w}) = \frac{exp(\bar{w} \cdot F(\bar{x},\bar{y}))}{\sum_{\bar{y}' \in y} exp(\bar{w} \cdot F(\bar{x},\bar{y}'))} \tag{2.11}$$

where the feature function described above is represented and the sum of all values over the y^n have been normalized.

Chapter 3

Related Work

This chapter discusses the research that has been conducted in the field of information extraction from product for e-commerce applications. The chapter is divided into several sections, aiming to group the different approaches to the problem and to provide an overview of the algorithms and techniques used for each different approach.

3.1 Datasets for e-Commerce

Natural language processing tasks are quite complex, with many particularities that require large amounts of previously noted knowledge to facilitate the task. Datasets like CoNLL2003, presented by (Sang and De Meulder, 2003) or OntoNotes 5.0 by (Pradhan et al., 2013), are great drivers in the scientific advancement of **NER** task.

(Uzuner et al., 2011) created I2B2 to challenge the scientific community to research and develop based on clinical records. The same happened when (Kim et al., 2003) created GENIA for the recognition of bio entities.

Researchers' interest in e-commerce has grown. Although there are several researchers from large corporations such as Amazon, eBay, Walmart, Rakuten, they do not practice to disclose the datasets used, making it impossible to reproduce or use new approaches to previously proposed methods.

Table 3.1 shows a summary of the datasets, the type of attributes present, the volume of data and where they can be found.

#	Dataset Name	Attributes Included	Sentences	Available at
1	Victoria's Secret and Others	product name, price, brand name, description, retailer, rating, style attributes, total and available sizes, color, ...	614,262	https://www.kaggle.com/PromptCloudHQ/innerwear-data-from-victorias-secret-and-others#amazon_com.csv
2	Electronic Products and Pricing Data	brand, category, merchant, name, prices, condition, source, ...	7,000	https://data.world/datafiniti/electronic-products-and-pricing-data
3	Men's Shoe Prices	brand, category, colors, description, features, ean, source, ...	10,000	https://data.world/datafiniti/mens-shoe-prices
4	Women's Shoe Prices	brand, category, colors, description, features, ean, source, ...	10,000	https://data.world/datafiniti/womens-shoe-prices
5	Best Buy E-commerce NER dataset	brand, category, model name, screen size, ram, storage, price	941	https://www.kaggle.com/dataturks/best-buy-ecommerce-ner-dataset
6	Amazon and Best Buy Electronics	brand, category, colors, name, reviews text, reviews title, reviews rating, ...	7,000	https://data.world/datafiniti/amazon-and-best-buy-electronics
7	Product details on Flipkart	url, name, category, description, brand, product specifications, ...	20,000	https://data.world/promptcloud/product-details-on-flipkart-com
8	Fashion products on Amazon	name, manufacturer, price, number of reviews, average review rating, customer review, category, description, product information, ...	22,000	https://data.world/promptcloud/fashion-products-on-amazon-com
9	Abt-Buy	name, description, price, manufacturer	2,173	https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution(Köpcke et al., 2010)
10	Amazon-GoogleProducts	name, description, price, manufacturer	4,589	https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution(Köpcke et al., 2010)
11	Walmart	brand, upc, title, price, short description, long description, dimensions, ...	2,554	http://pages.cs.wisc.edu/~anhai/data/corleone_data/products/(Gokhale et al., 2014)

TABLE 3.1: A summary of public e-Commerce datasets.

Although there are several datasets previously shown by Table 3.1, most of them have missing values or bad catalog values.

Several researchers from large corporations such as eBay (Putthividhya and Hu, 2011), Walmart (More, 2016)(Zhang and LeCun, 2015) and Rakuten (Shinzato and Sekine, 2013) use huge datasets, different from those previously presented. They do not have the practice of disseminating the datasets, which makes their reproduction or new approaches to previously proposed methods impossible.

3.2 Attribute-Value Pairs Extraction

The **AVP** extraction is the problem of identifying the values for one or more attribute of any entity and many researchers have fallen into this field of investigation.

(Ghani et al., 2006) uses a supervised model to extract the **AVP** through Naive Bayes (**NB**). To become independent of the annotated data, they inserted a

semi-supervised layer co-Expectation-maximization over the **NB**, making it possible to scale the model using a limited set of labeled training data.

(Raju et al., 2009) presented an unsupervised approach to extract product attributes using ngrams to calculate the similarity between noun phrases that later is used by a clustering algorithm. Attributes are obtained through a ranking function. Although they also use ngrams, (Putthividhya and Hu, 2011) generate their training dataset via bootstrap from matching n-grams words, with dictionaries and posteriorly manually inspected to guarantee that they have no flaw. The result is a supervised named entity recognition pipeline that extracts attributes from product title.

(Kovelamudi et al., 2011) proposes a supervised system to extract the attributes of the products through the reviews made by the customers to them. To this end, they created a database of semantic relationships where they correlate the words highlighted in customer reviews and articles in Wikipedia or on the Internet. They use the support vector machine to train the model using the hand-crafted features previously highlighted. Despite using customer reviews, (Broß and Ehrig, 2013) adopted an unsupervised approach were their system depends on heuristic filtering to obtain candidates from customer reviews. Highlight that sentiment expressions are detected through a hand-crafted compound lexicon. All terms referring to the product or brands are removed through a stop word list previously created.

(Shinzato and Sekine, 2013) applies an unsupervised model to automatically create a Knowledge Base (**KB**) from product pages tables. Having already built a **KB**, they used it to create a new set of annotated data. The model chosen was the **CRF** and an **AVP** layer is used for each category. Similar to the previous approach, (Charron et al., 2016) used consumer patterns along with subtrees-based extraction and information listing to create the annotation of data-driven products. It is an end-to-end unsupervised architecture.

Having made a similar approach to (Shinzato and Sekine, 2013), (Bing et al., 2016) proposed to extract attribute value in an unsupervised way that was trained by a hidden CRF model. Their method uses latent Dirichlet allocation (Blei et al., 2003), deconstructing the sentence and finding the unknown concepts crucial to popular features, assigning to a domain.

3.3 Neural Sequence Labeling Models

NER is a close research area to our problem and has been widely addressed in scientific literature. While the first systems for recognizing names were based on pattern matching rules and pre-compiled lists of information, the research community has since moved towards employing machine learning methods for creating such systems (Goyal et al., 2018).

Models used for predicting Named entities in text sequence faced the structured prediction problem and can be broadly classified into supervised, semi-supervised and unsupervised models.

The supervised learning requires a lot of annotated data and the costs of their creation contribute to choose alternative learning methods.

Semi-supervised is an alternative that needs a small labeled training set and a huge corpus of unlabeled data.

Unsupervised learning is the opposite of the supervised learning approach and is also an alternative. It works without any label data and its task is to find patterns in the unlabeled data. Below, we detailed some approaches.

(Huang et al., 2015) is a pioneer in the implementation of the **BiLSTM-CRF** model, where it represents each word of the sentence vectorially, and through a hand-craft rules system, the vector goes to the **CRF** layer. The hand-craft rules system is to handle spelling and context features like uni-grams, bi-grams and tri-grams.

(dos Santos and Guimarães, 2015) proposes a deep neural network that uses char embeddings and word embeddings jointly as input to the convolution layer. They framed their approach was a sequential classification problem. It also uses a dropout layer on **BiLSTM** output nodes to reduce model overfitting. The output as normalized after the Viterbi layer found the most probably tag for word. (Chiu and Nichols, 2016) also used a hybrid model where **CNN** receives the junction between character and word embedding. **BiLSTM** applies each iteration with a linear layer and a softmax layer to calculate the log probabilities of each tag category. It also uses a lexicon with the BIOES tag as an annotation.

(Lample et al., 2016) consists in a **BiLSTM-CRF** model. The **CRF** layer

is responsible for outputting the correct tag by maximizing the matrix transition scores. Similarly, (More, 2016), (Rei, 2017), (Ma and Hovy, 2016) and (Yadav et al., 2018) propose an identical system where (More, 2016) use distant supervision technique with a rule-based strategy to obtain the dataset. Also, normalize at the end to ensure the existence of unique value. (Rei, 2017) present a **BiLSTM** capable of multitasking and encouraged to discover new features and uncheck an "O" tag so as not to submit the template. Additionally, (Ma and Hovy, 2016) present a **CNN** layer similar to (Chiu and Nichols, 2016) except that the input data of the model presented by them are only character embeddings and these use a junction char + word embeddings. It also applies a dropout layer to the data input on **CNN**. (Yadav et al., 2018) shown a similar approach, but learns the structure and their n-gram suffix and prefix of each word.

(Dirie, 2017) proposed a similar approach to (Yadav et al., 2018), where he addresses character-level and word-level with **BiLSTM** and word-embedding trains him in an unsupervised way through skip-gram. In the last layer, it applies a **CRF** Multiplex layer which allows an attribute-by-attribute tagging policy capturing previously unknown attributes more efficiently.

(Shen et al., 2017) aims to reduce the amount of tagged data required to train the model while maintaining its quality. For this, it uses deep learning along with active learning. In order not to overload the system with model retraining, it adds the new data along with the old one and updates the neuronal network weights to a lower number of epochs. The active learning layer select sentences that have been predicted with the lowest normalized size log probability. The **CNN-CNN-LSTM** architecture works with a character-level convolutional encoder, another convolutional encoder for words, and the final layer is **LSTM** as a decoder tag.

(Zheng et al., 2018) frames the problem as a sequence tagging task where it uses a **BiLSTM-CRF** attention model. The tagging strategy adopted is similar to (Lample et al., 2016). It uses an active learning strategy using the flip method tag to drastically reduce the need for manually annotated data.

Authors	Features	Architecture Resume	Structure Tagging	Embeddings	Datasets Used
(Huang et al., 2015)	Yes	BiLSTM output vector + hand-craft rules features vector connected to CRF	CRF	(Collobert et al., 2011) pre-trained with 50-dimensions	CoNLL2000 + CoNLL2003
(dos Santos and Guimarães, 2015)	Yes	char-level + word-level embeddings as input to CNN, minimize the negative log-likelihood with stochastic gradient descent	Sentence-level log-likelihood	pre-trained word embeddings Skip-gram + char-level embeddings extracted with a CNN	SPA CoNLL2002 + HAREM I
(Chiu and Nichols, 2016)	Yes	char + word embeddings as input to BiLSTM layer, BiLSTM output with dropout decoded by linear layer + log-softmax layer into log-probabilities for each tag category	Sentence-level log-likelihood	(Collobert et al., 2011) pre-trained with 50-dimensions + char-level embeddings extraction with a CNN	CoNLL2003 + OntoNotes 5.0/CoNLL2012
(Lample et al., 2016)	No	char + word embeddings as input to BiLSTM layer, CRF receives the output vector to decode label sequence	CRF	pre-trained word embeddings skip-n-gram + char-level embeddings extraction with BiLSTM	CoNLL2002 + CoNLL2003
(Rei, 2017)	No	BiLSTM with additional language modeling to predict sequence label using softmax layer as output.	Sentence-level log-likelihood	pre-trained word embeddings word2vec with 300-dimensions + trained word embeddings PubMed + PMC with 200-dimensions	FCE + CoNLL2014
(Ma and Hovy, 2016)	No	jointly char + word embeddings as input to BiLSTM layer, CRF receives the output vector to decode label sequence	CRF	pre-trained word embeddings Glove with 100-dimensions + char-level embeddings extracted with CNN	WSJ + CoNLL2003
(Yadav et al., 2018)	Yes	char-level embeddings as input to BiLSTM layer + BiLSTM output vector with word-level embedding as input to BiLSTM + CRF input fedded by BiLSTM word-level output vector	CRF	char-level embeddings extraction with BiLSTM + pre-trained word embeddings Fasttext with 300-dimensions + pre-trained word embeddings Glove with 100-dimensions + pre-trained word embeddings PubMed with 300-dimensions	CoNLL2002 + CoNLL2003 + DrugNER (MedLine + DrugBank) + I2B2
(Dirie, 2017)	No	char-level embeddings as input to BiLSTM layer + BiLSTM output vector with word-level embedding as input to BiLSTM + CRF Multiplex input fedded by BiLSTM word-level output vector	CRF	char-level embeddings extraction with BiLSTM + trained word embeddings Skip-gram	Rakuten
(Shen et al., 2017)	No	CNN output vector as input jointly with word embedding into CNN + LSTM receives CNN output vector and decode label sequence through LSTM layer + active learning to help reduce reliance on tagged training data	Sentence-level log-likelihood	char-level embeddings extraction with CNN + trained word embeddings word2vec	OntoNotes 5.0 + CoNLL2003
(Zheng et al., 2018)	No	BiLSTM-CRF with attention model + active learning using flip tag method	CRF	pre-trained word embeddings Glove with 100-dimensions	Amazon

TABLE 3.2: A summary of neural sequence labeling models.

Chapter 4

Attribute-Value Pairs Extraction

Following the related work and taking into account the investigation methodology used, it follows the stage of developing a solution considering all the facts previously reported.

Figure 4.1 shows the pipeline used in our model and all the steps required to develop it.

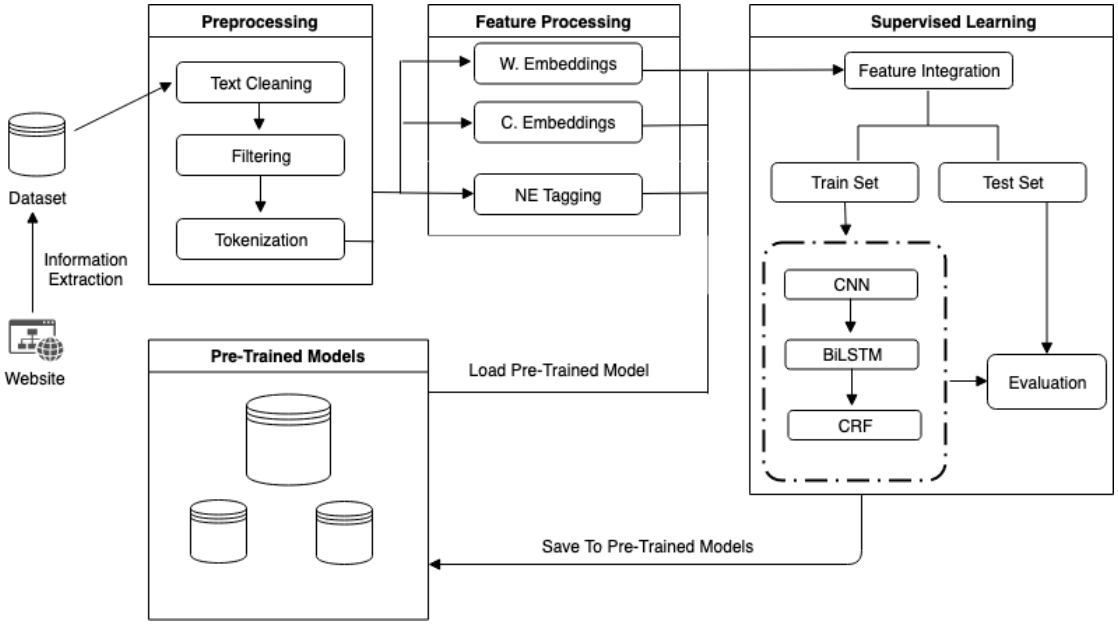

FIGURE 4.1: Pipeline of our approach.

In the following sections you will find the explanation of each step and the reason for the decisions taken.

4.1 Datasets

Due to the difficulty in getting an e-Commerce dataset that contained the information about the products (e.g. product title or product description) properly labeled with the part-of-speech or/and named entities tag, we felt the need to create a dataset from scratch.

The creation of the datasets belongs to the field of research Web Data Extraction where the structure of the HTML mark-up language present in the web pages was used as an advantage for the extraction.

Not being the focus of this project and after a brief literary review on the theme in question (Ferrara et al., 2014), we decided to use as a technique of information extraction a tree-based approach.

The DOM (Document Object Model) is the representation of the HTML pages in plain text, where the elements are represented through the HTML tags as well as the text present on each page. The tree structure is easily understood, with each HTML tag representing a node. These structures are called DOM Tree.

The technologies used in the development of this artifact to obtain the necessary data for the construction of the dataset was the java programming language, using the Selenium library[1] to interact with the web page in an automated way. Apache POI[2] was also used to create, write and modify the XLS/XLSX spreadsheets to save all information extracted.

Being our focus on the e-Commerce websites, where the product listings are structured dynamically, it is necessary to use the existing structural similarity in the DOM trees to extract efficiently and effectively without human iteration.

The product listings are represented in lists, where the iteration with the various products is done through an iterator. At each iteration, all the information about the attributes and values of each product is obtained through the tables. The change of page for a new listing is made when there are no more products in the current list.

[1] https://www.seleniumhq.org/download/
[2] http://poi.apache.org/download.html

We created three new datasets, two belong to the fashion sector and the other is a junction of the sectors: home, furniture, appliances, sports, fitness and outdoors.

In order to ensure the confidentiality of the websites involved in this process, we will now call the dataset with limited data of Jewels (2214 products), the dataset that contains a large amount of products of Fashion (21775 products) and the dataset that is from a different sector than fashion of HomeDecor (250 products).

4.1.1 Datasets Structure

We decided to opt only for the inclusion of product descriptions because they already contain the information on the title and are an aggregator with more information that can be extracted.

We created three datasets with the concept of testing the generalization between datasets of different categories and sizes without having been previously trained.

Jewels dataset is rather limited, contains only and exclusively 2214 products related to the jewelry industry where the descriptions are mostly structured (Figure 4.2).

The Casio Edifice brand men watch from the Retrograde Chronograph collection with the EFV-530GL-5AVUEF reference.
The watch is water resistant to 10 ATM / 100 M , the value in meters does not refer to diving depth, but to the air pressure used during the water resistant test.
The genuine leather strap - The high quality genuine leather in brown color wrist strap combines durability with style and gives your watch a classic look.
The watch also has an brown 47.2 dial and rose gold as the case color.

FIGURE 4.2: Example of a product description from Jewels dataset.

In creating the Fashion dataset, we paid attention to the two important particularities that we wanted to achieve. A very significant number of descriptions compared to Jewels dataset and the insertion of three new categories within the same sector to perform correlation tests.

Please note in Figure 4.3 that the product descriptions of the Fashion dataset come from the brands themselves. It is notorious the difference in structuring compared to Jewels, where the raw text of the Fashion will be an added value for the diversification of the model.

Versace's collaboration with motor company Ford was inspired by 'the excitement of buying your first car'. They utilise the signature Ford logo to reflect emotions of joy and thrill. Made in Italy, this back cotton x Ford logo print hoodie from Versace features a hood with a contrast printed logo in the rear, dropped shoulders, long sleeves, a central printed Ford logo, a kangaroo pocket and a ribbed hem and cuffs.

FIGURE 4.3: Example of a product description from Fashion dataset.

The HomeDecor dataset was created with the perspective of being the most different from Jewels and Fashion and in its genesis is the concern of the diversity of the tests, with the vast majority of its 250 products being disparate and different from the fashion sector.

The descriptions of their products are unstructured and with fewer standards of the three datasets in question. Figure 4.4 visually demonstrates what we explained above.

Transform your bedroom with this Shalini Diamond Stitched Platform Bed by Zinus. Classic styling and strong reliable wood slat support for your spring, memory foam, latex, or hybrid mattress. Ships in one carton with the frame, legs and wooden slats conveniently located in the zippered compartment in the back of the headboard for easy assembly. No box spring needed. Featuring a classically styled headboard and low profile footboard frame this platform bed offers strong, reliable support for your mattress.

FIGURE 4.4: Example of a product description from HomeDecor dataset.

This dataset was conceived for inference task, where the main objective is to known new words and recognize those already known.

4.1.2 Annotation

After the extraction of the information and the creation of the dataset, it is now important to proceed with the labelling of the data taking into account the custom named entities.

Based on the related work, we adopt the distant supervision that will be explained below.

Distant Supervision Model

Due to limited human capacity and the high annotation costs of large datasets, researchers have felt the need to adopt new approaches and techniques for rapid, effective and low false-positive labeling.

We adopted the use of the distant supervision model after reading several identical projects(Dirie, 2017).

The distant supervision model starts with the existence of datasets or external sources of information(Mintz et al., 2009; Shinzato and Sekine, 2013; Wu and Weld, 2007) and uses them to complement the relationships needed to label the dataset, thus making it possible to have a large set of training data, guided in the annotation process.

Assuming that the sites where the information was extracted contained the correct information about the value attribute pairs in their tables, as shown in Figure 4.5 and Figure 4.6, we used our previously created csv file as a reliable external source to guide the annotation.

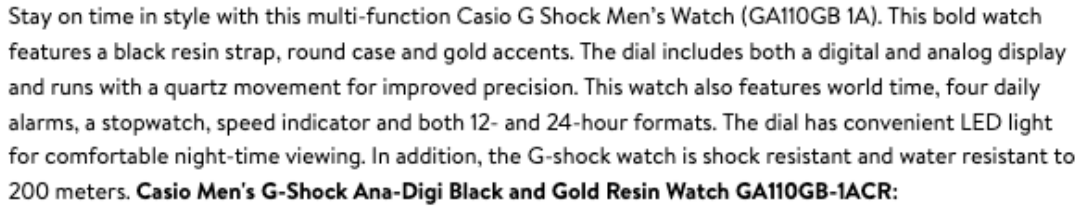
Stay on time in style with this multi-function Casio G Shock Men's Watch (GA110GB 1A). This bold watch features a black resin strap, round case and gold accents. The dial includes both a digital and analog display and runs with a quartz movement for improved precision. This watch also features world time, four daily alarms, a stopwatch, speed indicator and both 12- and 24-hour formats. The dial has convenient LED light for comfortable night-time viewing. In addition, the G-shock watch is shock resistant and water resistant to 200 meters. **Casio Men's G-Shock Ana-Digi Black and Gold Resin Watch GA110GB-1ACR:**

FIGURE 4.5: Example of a product description extracted from product page

For each previously extracted product description, a word-by-word annotation is made with the attribute values from the product table. Whenever the description word and attribute value are equal, the word is tagged with "B-*" and followed by the tag, indicating the beginning of this attribute. If the attribute value contains more than one word, this indicates that the next word in the description also has the same attribute. Since the previous word is of the same attribute, we use "I-*" as the prefix followed by the tag. All other words in the description that are not equal to any attribute values are labeled "O".

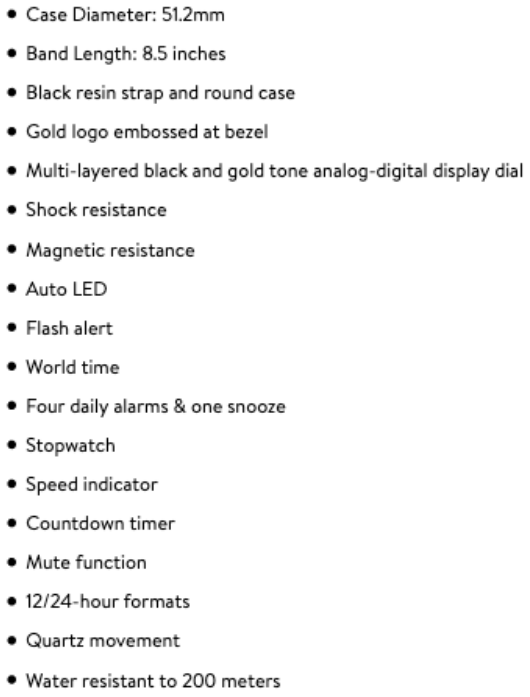

FIGURE 4.6: Example of a attribute-values extracted from the product page

4.1.3 Custom Entity Tagging

In Section 2.2.3, we introduce of the named entity recognition task where we described its importance in the recognition of entities as well as the possible annotation schemes.

The labels appear as a facilitator in the task of recognizing entities present in the input text and the entities present in e-Commerce websites are quite different from the entities present on CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003).

The difference in contexts leads us to create our entities to correctly label the products.

As mentioned in Section 2.2.3, the tag scheme is very important in the recognition of entities. According to (Reimers and Gurevych, 2017), BIO stands out and is the most recommended choice for tasks related to entities.

Table 4.1 shows an example of the BIO annotation scheme that we adopted to encode the different entity tags, using the example of Figure 4.7.

Important to realize that entity tags allow values of type strings and integers.

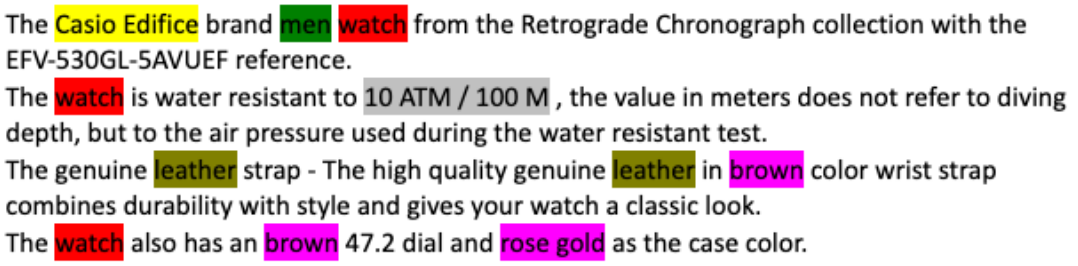

FIGURE 4.7: Example of product description with custom entities tags.

Tag	Label Meaning	Example Given
B-B	Beginning of a Brand	Casio
I-B	Inside of a Brand	Edifice
B-G	Beginning of a Gender	men
B-PC	Beginning of a Product Color	brown
B-PC	Beginning of a Product Color	rose
I-PC	Inside of a Product Color	gold
B-PM	Beginning of a Product Material	leather
B-PT	Beginning of a Product Type	watch
B-WR	Beginning of a Water Resistance	10
I-WR	Inside of a Water Resistance	ATM
I-WR	Inside of a Water Resistance	/
I-WR	Inside of a Water Resistance	100
I-WR	Inside of a Water Resistance	M

TABLE 4.1: Custom Entity Tag with BIO tag scheme using Figure 4.7 as example

Quality Concerns

Tasks such as data tagging are undergoing this change due to the difficulty in arranging large datasets to train natural language models.

The issue of quantity for quality is well suited to the problem.

Our approach to the problem was initially via distant supervision and then we manually check the quality of the data for possible annotation errors.

4.2 Supervised Learning Model

After a brief general introduction and visualization of the general pipeline of the artifact (Figure 4.1), we will now discuss in detail the algorithms used as well as their main features.

We can see the structure of our model as being divided into three layers, where the first layer is **CNN**, the second is **BiLSTM** and the last is **CRF**.

We used a CNN-BiLSTM-CRF that is capable of transfer learning based on (Ma and Hovy, 2016) approach where Figure 4.8 exhibits the interaction between the algorithms.

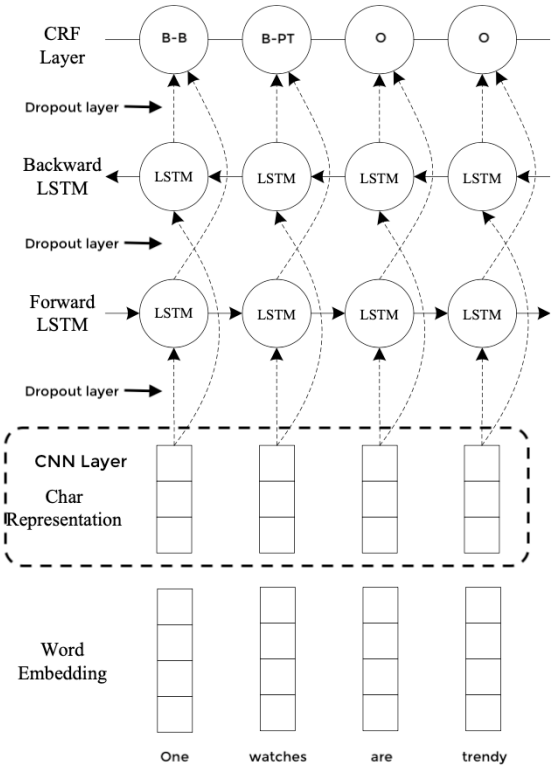

FIGURE 4.8: Architecture of our neural network (Ma and Hovy, 2016).

4.2.1 Convolutional Neural Network

After a brief introduction of the operation of the **CNN** algorithm in Section 2.4, where convolution and pooling concepts have been demonstrated and explained, we intend with this chapter to describe in detail how and why the **CNN** algorithm is being used in **NLP** tasks and its relevance in the current state of the art.

Our approach contemplates a convolution and max pooling layer that tries to capture the spelling and morphological characteristics of words or words in the context of the sentence, thus allowing rare or misspelled words as well as prefixes and suffixes words to be recognized through the use of language models based only on the spelling of the word and its similar vectors (Gridach, 2017). This character-level layer and can be LSTM-based or CNN-based. We choose a CNN-based approach according to (Zhai et al., 2018, p.38), "the models using CNN-based character-level word embeddings have a computational performance advantage, increasing training time over word-based models by 25% while the LSTM-based character-level word embeddings more than double the required training time."

Figure 4.9 illustrates how the text is represented as a matrix where each row of the matrix corresponds to a character.

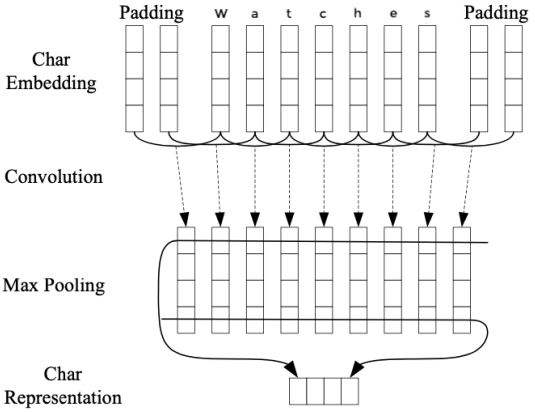

FIGURE 4.9: Character-level information encode into Convolutional Neural Network (Ma and Hovy, 2016)

The efficiency in terms of representation makes convolution filters achieve good representations automatically without having to represent the entire vocabulary. Thus, it can capture the features similarly to n-grams but representing it in a more compact way and without suffering from its limitations.

Word Embeddings

We have chosen not to use an unsupervised word embedding for the simple reason that unsupervised models are only adequate if the dataset used for training has an appropriate size.

The word embeddings chosen were trained using structural information from dependency graphs and according to (Komninos and Manandhar, 2016, p.1498), "the dependency-based word embedding largely improved the performance for semantic relation identification".

The character embeddings + word embeddings approach used by us after (Ma and Hovy, 2016) has empirically been shown that will bring better results to the final model.

Considering the points above, the concatenation between the embeddings generated from character-level with word embeddings are the input of the **BiLSTM** layer.

4.2.2 Bidirectional LSTM (BiLSTM)

BiLSTM is the extension of the **LSTM** referenced in Section 2.4. It uses past information as well as future information to predict.

In other words, **BiLSTM** is a set of two layers of **LSTM** where one layer processes the information from left to right (forward) and the other layer, from right to left (backward).

Figure 4.10 shows the structure of **BiLSTM**.

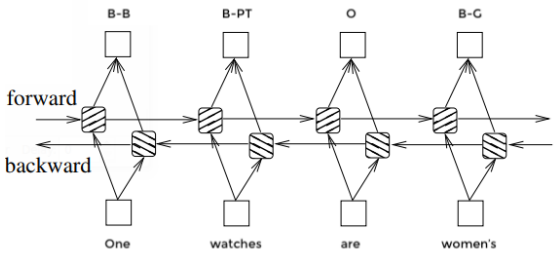

FIGURE 4.10: Architecture of Bidirectional Long-Short Term Memory. Adapted from http://colah.github.io/posts/2015-09-NN-Types-FP/

4.2.3 Conditional Random Fields (CRF)

After a brief introduction of the **CRF** equations in Section 2.5, where we present the feature function and the linear-chain **CRF**, we will now present the full expanded linear-chain **CRF** in Equation 4.1.

$$P(\bar{y}, \bar{x}; w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \bar{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \bar{x}, i))} \tag{4.1}$$

where $\sum_i$ is the length of sequence x, $\sum_j$ is the sum over all feature function, w_j is the weight for given feature function and $\sum_{y' \in Y}$ is the sum over all possible tag sequence.

The previous equation is represented globally by the feature function f_k by k, where it makes the sums of all the features functions by the different n transition states existing in $\bar{y}$. Thus, the entire sequence is mapped in $F(\bar{x}, \bar{y}) \in \mathbb{R}^d$

The **CRF** is the last layer of the model and this makes it possible to add constraints to the final labels to ensure their validity.

The constraints are automatically applied and learned, based on the set passed during the training phase.

The **CRF** layer receives an matrix (number of words in a sentence multiplied by the number of possible labels for each word) with the P_{ij} results where j^{th} is the probability that the tag is the correct i^{th} word in the sentence. For better understanding, Figure 4.11 represents the dynamics between the **BiLSTM** and **CRF** layers, matrix with the probabilities and the **CRF** layer giving the output results that maximize the score function.

There is also a transition matrix that represents the probability of transition between tags. It allows us to get information about the veracity of the model, conclude unlikely transitions as well as outliers. An example is that transitions from I-* to B-* are never possible in practice and if the weights are found to be positive and high, it is a sign of incongruity in the model.

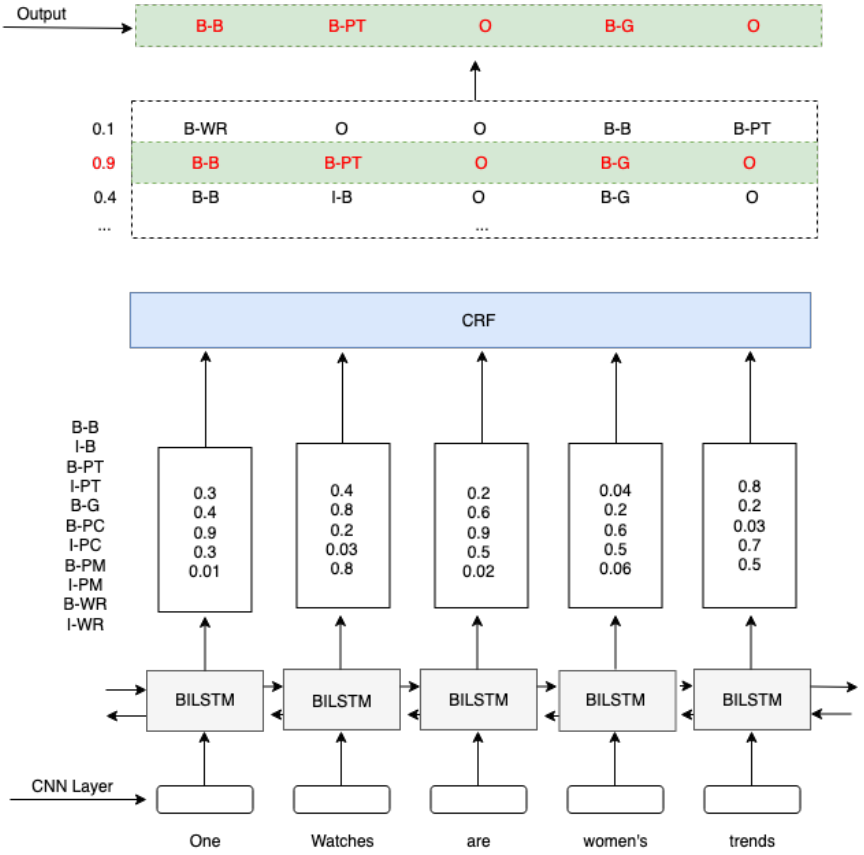

FIGURE 4.11: Emission score from BiLSTM layer. Adapted from https://createmomo.github.io/2017/09/23/CRF_Layer_on_the_Top_of_BiLSTM_2/

The score function for predicting a sequence is given by the sum of the transition between the current state tag and the next state tag, along with the probability from **BiLSTM** of being the correct tag for the current word as shown in Equation 4.2.

$$P_{score}(y) = \sum_{i=0}^{n} T_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{4.2}$$

which is then used in Equation 4.3 to find a single tag for each word so that the junction of all the single tags gives the highest value of the joint probability.

$$argmax_y p(y|X; P_{score}(y)) \tag{4.3}$$

4.2.4 Summary

Our approach takes into account the approaches presented in the related work (Ma and Hovy, 2016; Dirie, 2017). The structure is **CNN-BiLSTM-CRF** where **CNN** deals with the character-level that together with the word-embedding is the input of the **BiLSTM** layer, that through its powerful ability to analyze sentence sequences has as output the matrix of probabilities of tags for each word. Also, **CRF** that together with the emission matrix received from **BiLSTM**, gives the tag sequence that maximizes the score function.

Chapter 5

Experimental Settings and Evaluation

5.1 Experimental Settings

In this chapter we explain in detail the composition of the datasets, the metrics used to evaluate the model, and the components required to run the model.

As described in Section 4.1, we created three datasets to test the generalization capability of the model.

Table 5.1 shows the structure of each dataset as well as the categories it contains and their size.

Dataset	Categories	Size
Jewels	Jewelry	2214
Fashion	Clothing, Shoes, Accessories & Jewelry	21775
HomeDecor	Home, Furniture & Appliances + Sports & Outdoors	250

TABLE 5.1: Datasets Structures

For a better perception of the diversity between types of products within the dataset, we present the following figures.

Figure 5.1 and Figure 5.3 show that there is a lot of data disparity between certain types of products, however, it was made purposely to realize how the system interacts and reacts to this type of adversities.

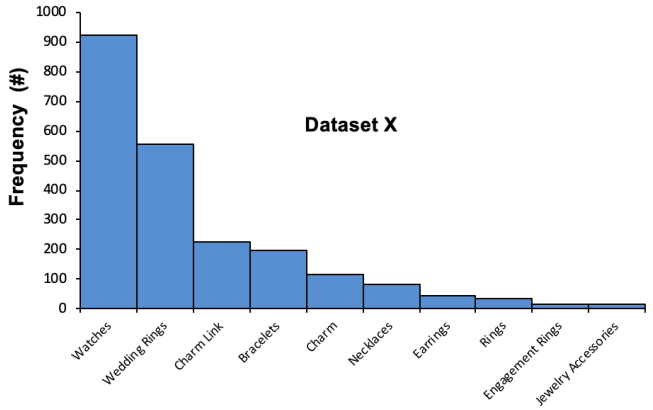

FIGURE 5.1: Distribution by product type of the Jewels dataset

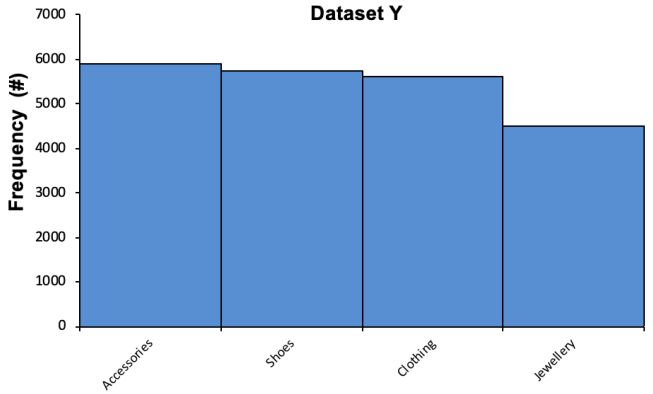

FIGURE 5.2: Distribution by product type of the Fashion dataset

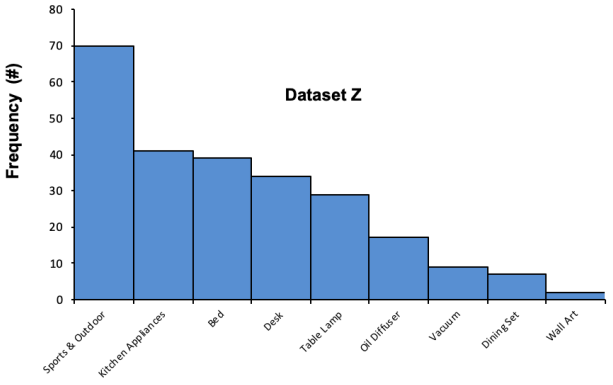

FIGURE 5.3: Distribution by product type of the HomeDecor dataset

In this project we focus on only 6 classes and 11 labels. Note that in Figure 5.4 does not appear water resistant because this class only belongs to the Fashion dataset, as such, there would be no comparison term. We do not use the I-G tag because there is no genre with more than one word. Figure 5.4 shows the distribution between the various classes by the 3 dataset.

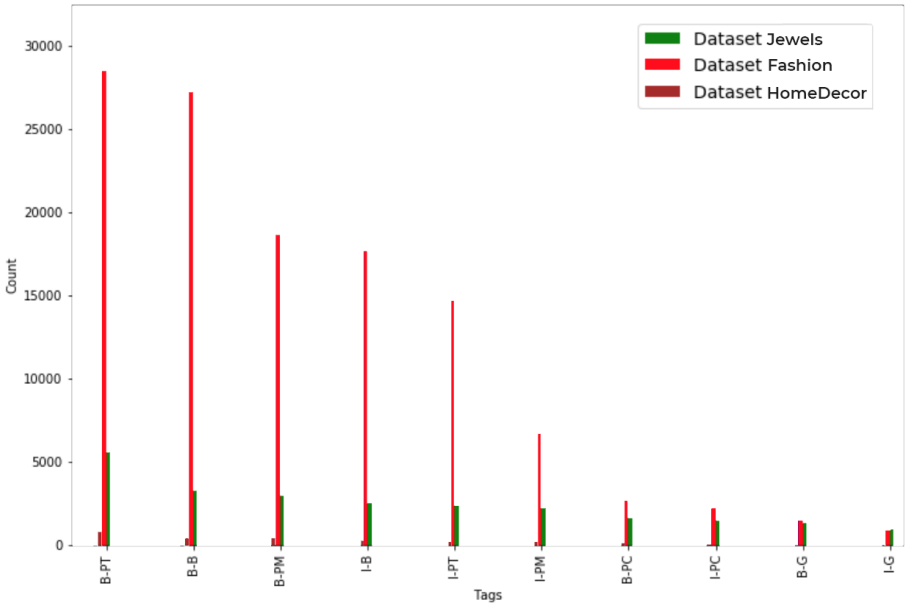

FIGURE 5.4: Distribution of dataset tags

Through the image, we observe that the dataset Fashion and Fashion+ contain much more data and their distribution prevails in the classes B-PT and B-B.

Our tests will use the three most commonly used metrics: Precision, Recall and F1-Score. Table 5.2 represents the meaning of True Positive, False Positive, and False Negative in the context of entity recognition.

Metrics	Meaning in this context	Gold Label	Predict Label
True Positive (TP)	Token and predicted token are positive	B-B	B-B
False Positive (FP)	Token are negative but predicted token is positive	O	B-B
False Negative (FN)	Token are positive but predicted token is negative	B-B	O

TABLE 5.2: Meaning of metrics

True negative happens when the token and its prediction are both negative. In this case, tags being equal means they are well labeled and its fits True Positive, however, due to the high hit rate and the direct relationship with the input data, we chose not to sum it to make the results more realistic.

The precision formula is Equation 5.1, the recall is Equation 5.2 and the f1-score in Equation 5.3.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

5.1.1 Experimental Setup

Our model was made in the python[1] programming language in version 3.6 on Ubuntu 18.04[2] operating system. We use keras 2.2.0[3] and tensorflow 1.8.0[4] as backend.

One of the initial problems felt was the incompatibility of library versions together with Ubuntu and CUDA drivers[5], and it is really necessary to be aware of version compatibility when creating or reusing an existing model. We use one GeForce GTX 1080 Ti to run the model.

Model configuration hyperparameters are very important and have a real impact on model evaluation. In order to maximize performance, we use the best settings for sequential labeling tasks, according to (Reimers and Gurevych, 2017).

Table 5.3 shows a list of the hyperparameters used by model.

Parameter	Selected Value
Dropout	0.25
Classifier	CRF
LSTM-Size	100
Optimizer	Adam
miniBatchSize	32
charEmbeddings	CNN
charEmbeddingsSize	30
charFilterSize	30
charFilterLength	3

TABLE 5.3: Hyperparameters used in the model

[1] https://www.python.org/downloads/
[2] https://ubuntu.com/download/desktop
[3] https://pypi.org/project/Keras/2.2.0/
[4] https://pypi.org/project/tensorflow/1.8.0/
[5] https://developer.nvidia.com/cuda-92-download-archive

5.1.2 Results

This section contains the results of the experiments and an explanation of the limitations of our model for future work.

Our first approach was to create the Jewels dataset to realize the scalability of the system, training with a small dataset and realize its generalization to a larger one.

The distribution of data by categories as well as by labels was taken into account in the creation of the Fashion dataset, making it a balanced dataset.

For the purpose of predicting attributes and values, we create the HomeDecor dataset exclusively for testing. As mentioned earlier, the product categories are quite diverse and no pre-processing was applied to the data.

To try to understand the effect of a small dataset on a large one, we put together the dataset Jewels + Fashion for testing purposes. We call it Fashion+.

The Table 5.4 shows the results of the dataset Jewels, Fashion, and Fashion+. The data previously took into account 5-fold cross validation.

Dataset	F1-Score
Jewels	99.68
Fashion	94.42
Fashion+	95.34

TABLE 5.4: F1-score comparison of the different datasets.

5.1.3 Inference Tests

Table 5.5 shows the results of the Jewels dataset inference when training was performed by the Fashion dataset.

Table 5.6 shows the results of the HomeDecor dataset inference when training was performed by the Fashion+ dataset.

Table 5.5 and Table 5.6 show that attributes with a limited number of possible values such as product color, material, water-resistance, and gender are easily generalizable whatever the data set category, with high hit rates.

	precision	recall	f1-score	support
B-B	0.56	0.37	0.44	2951
B-G	1.00	1.00	1.00	2236
B-PC	0.82	0.81	0.81	2988
B-PM	0.42	0.41	0.42	2521
B-PT	0.73	0.67	0.69	6277
B-WR	1.00	0.80	0.89	924
I-B	0.97	0.85	0.90	612
I-PC	0.52	0.45	0.49	1349
I-PM	1.00	0.40	0.57	1472
I-PT	0.00	0.00	0.00	1628
I-WR	1.00	0.28	0.44	3265
O	0.94	0.99	0.97	117698
accuracy			0.91	143921
macro avg	0.75	0.59	0.64	143921
weighted avg	0.90	0.91	0.90	143921

TABLE 5.5: Jewels dataset deduced from Fashion dataset.

	precision	recall	f1-score	support
B-B	0.28	0.19	0.23	436
B-G	0.80	0.92	0.86	13
B-PC	0.82	0.79	0.80	127
B-PM	0.87	0.43	0.58	390
B-PT	0.35	0.04	0.07	809
I-B	0.45	0.24	0.31	307
I-PC	0.65	0.38	0.48	45
I-PM	0.96	0.16	0.28	158
I-PT	0.00	0.00	0.00	213
O	0.94	0.99	0.97	30889
accuracy			0.93	33387
macro avg	0.56	0.38	0.42	33387
weighted avg	0.91	0.93	0.91	33387

TABLE 5.6: HomeDecor dataset deduced from Fashion+ dataset.

Jewels inference results compared to HomeDecor are better due to training of Fashion dataset containing 20.67% of products in the jewelry category while Fashion+ does not contain any category or attribute similar to those in the HomeDecor dataset.

To understand the quality of the inference by our model, we made a comparison between the gold label and the prediction through Table 5.7 and Table 5.8, where it is presented a top 5 of the most used words for each attribute.

Tag	#	Gold		Predicted	
	1	Lotus	630	M	781
	2	One	602	One	602
B-B	3	Eternis	552	Nomination	236
	4	Nomination	472	Anjewels	84
	5	Anjewels	168	Lotus	71
	1	Jewels	482	Jewels	482
	2	Edifice	94	Colors	36
I-B	3	Colors	36	Jewelry	16
	4	-	-	Gun	1
	5	-	-	-	-
	1	watch	3700	watch	2776
	2	wedding	1100	ring	1210
B-PT	3	Charm	682	Charm	682
	4	bracelet	398	case	437
	5	necklace	167	bracelet	398
	1	ring	1144	-	-
	2	Link	452	-	-
I-PT	3	Accessory	32	-	-
	4	-	-	-	-
	5	-	-	-	-
	1	316L	588	steel	1240
	2	gold	552	leather	632
B-PM	3	leather	322	stainless	589
	4	genuine	262	Resin	3
	5	Rubber	60	-	-
	1	stainless	588	steel	589
	2	steel	588	-	-
I-PM	3	leather	264	-	-
	4	Rubber	2	-	-
	5	Silicone	2	-	-
	1	steel	515	black	547
	2	silver	505	silver	547
B-PC	3	black	475	white	382
	4	white	358	rose	344
	5	rose	277	gold	340
	1	gold	834	gold	1161
	2	&	228	&	4
I-PC	3	black	46	topaz	1
	4	rose	40	-	-
	5	brown	7	-	-
	1	5	656	5	656
	2	10	178	3	87
B-WR	3	3	87	-	-
	4	20	2	-	-
	5	9	1	-	-
	1	ATM	922	ATM	922
	2	M	781	-	-
I-WR	3	/	781	-	-
	4	50	558	-	-
	5	100	175	-	-

TABLE 5.7: Comparison between the gold label and the prediction - Jewels dataset deduced from Fashion.

Table 5.7 shows the weight of "Eternis" in attribute B-B and its correct inference would have an impact of 0.16 on the f1-score.

The same happens with the "ring", where it has a predominance over the I-PT attribute (70.27%). In this specific case, its non-inference is due to the fact that there are the "wedding ring", "ring" and "engagement ring" attributes and the non-recognition of the "wedding" as B-PT makes the ring that should be mostly classified as I-PT to be inferred as B-PT. Similarly, the same happens to

"stainless" because the model could not infer the "316L" as B-PM.

The "M" appears to be inferred from the "B-B" tag due to the existence of the M. Cohen and M Missoni brands in the training dataset. The inference of the correct label would be "I-WR" where this M represents the meters of water resistance depth.

As explained, these are not isolated cases and the mean macro result of 0.64 represents this. As such, it is a result to take into account, but it is easily improved using a larger and more diverse training dataset.

Tag	#	Gold		Predicted	
	1	Better	65	Zinus	22
	2	Mainstays	59	Better	20
B-B	3	Zinus	34	All	19
	4	Ozark	25	area	16
	5	Hamilton	20	Perfect	15
	1	Homes	65	&	20
	2	Gardens	63	Gardens	20
I-B	3	and	35	Homes	20
	4	&	31	Tent	3
	5	Trail	25	Bag	2
	1	bed	129	bag	16
	2	desk	118	lamp	16
B-PT	3	blender	44	top	15
	4	lamp	41	backpack	12
	5	bike	35	chain	6
	1	Lamp	61	holder	1
	2	Cooker	24	Bag	1
I-PT	3	maker	17	-	-
	4	Set	13	-	-
	5	Diffuser	11	-	-
	1	metal	67	metal	51
	2	stainless	35	stainless	31
B-PM	3	wood	33	steel	29
	4	glass	30	wooden	18
	5	steel	29	ceramic	16
	1	steel	33	steel	30
	2	,	20	leather	1
I-PM	3	and	19	-	-
	4	leather	14	-	-
	5	glass	13	-	-
	1	black	34	black	37
	2	white	23	white	26
B-PC	3	Gray	8	Gray	11
	4	grey	8	grey	9
	5	red	6	red	7
	1	and	8	and	13
	2	silver	4	black	2
I-PC	3	white	3	Light	2
	4	,	3	or	1
	5	black	3	silver	1

TABLE 5.8: Comparison between the gold label and the prediction - HomeDecor dataset deduced from Fashion+

As shown in Table 5.8, the model can detect new values such as "Zinus" and "Better" of attribute B-B. We found that 77.27% of the times that "Zinus" was identified derives from the previous word being "from".

As in the Jewels inference, the HomeDecor also obtained a bad inference from the model with an average macro of 0.42. In this case, it is not due to the high amount of values in specific attributes, but the difference of the existing categories between the datasets Fashion+ and HomeDecor.

Table 5.9 shows the numbers of inferred values, which are totally unknown to the model.

The non-interference of any value in attributes such as I-PM and I-PT demonstrates that punctuation and closed clauses words such as ",", "or" and "and" have an enormous impact on the correct classification of values. It is important to realize that it is only relevant in attributes that use more than one value to characterize as is the case of the product material, product color and water resistant.

		B-B	I-B	B-PC	I-PC	B-PM	I-PM	B-PT	I-PT	B-WR	I-WR
Z	Gold	63	30	4	4	11	12	36	15	-	-
	Predicted	25	6	1	0	5	0	7	0	-	-
X	Gold	11	3	3	1	7	4	4	2	5	6
	Predicted	7	2	0	0	2	0	1	0	2	1

TABLE 5.9: Numbers of new values inferred from datasets

All things considered, with access to large datasets and datasets for inference tests with more dispersed attribute-value but similar categories, we would easily get results similar to those presented by (Dirie, 2017).

5.1.4 Limitations

The results of the inference showed that our model has limitations in the recognition of values of shared attributes, such as colors and materials of products. Also, in transition words like “,”, “or” and “and”.

The problem is solved by increasing the dataset inserted in the model or changing the model architecture to active learning, taking away the explicit need for data.

Chapter 6

Conclusion and Future Work

In this paper, we propose a model capable of inferring value attributes through product descriptions.

Three new datasets were created using a tree-based extraction technique that, using the XML Language Path, was able to obtain the attributes-values of the tables as well as the description of the products. The datasets created were annotated using the distant supervision technique, where the difference in size and category diversity was a factor. The distribution of product types was created to have one or two predominant attribute types.

In this project were created 6 classes that originated 11 custom named entities to label the data coming from the three datasets created.

The use of the water resistant class increased the complexity when performing generalization tests. In the training dataset there were value attributes with the water resistant label and in the generalised dataset there was no such attribute.

The proposed model consists of **CNN-BiLSTM-CRF**, where **CNN** capture the morphological characteristics of words along with dependency based word embeddings allowed the sequential layers to infer new attributes-values previously unknown.

We performed 5-fold cross validation tests on the created datasets. Tests were also developed to verify the generalization capacity of the model.

As shown in Chapter 5, the model was able to infer with a macro average of

0.64, where 20.67% of the training dataset was in the same category as the tested dataset. It was also tested on a dataset where the categories were totally different and obtained a result of 0.42 macro average.

Due to the different context of the tests, it is not possible to make a direct relationship between the results obtained and the state-of-the-art results.

The model was case sensitive and the generalization between attributes with limited number of values was effective. Transition words place obstacles in the right inferences.

As future work, we propose two different approaches:

- Transfer learning from one dataset in english to another dataset in another language using shared embeddings.
- Explore unsupervised approaches in this type of tasks.

Bibliography

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Baymard (2019). E-Commerce Product Lists Filtering Usability: An Original Research Study - Product Lists Filtering - Baymard Institute. (Accessed: 31-Jul- 2019).

Bing, L., Wong, T.-L., and Lam, W. (2016). Unsupervised extraction of popular product attributes from e-commerce web sites by considering customer reviews. *ACM Trans. Internet Technol.*, 16(2):12:1–12:17.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Broß, J. and Ehrig, H. (2013). Terminology extraction approaches for product aspect detection in customer reviews. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 222–230, Sofia, Bulgaria. Association for Computational Linguistics.

Camacho-Collados, J. and Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *J. Artif. Int. Res.*, 63(1):743–788.

Charron, B., Hirate, Y., Purcell, D., and Rezk, M. (2016). Extracting semantic information for e-commerce. In *International Semantic Web Conference*, pages 273–290. Springer.

Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dirie, A.-H. A. (2017). *Extracting diverse attribute-value information from product catalog text via transfer learning*. PhD thesis, Massachusetts Institute of Technology.

dos Santos, C. and Guimarães, V. (2015). Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33, Beijing, China. Association for Computational Linguistics.

Ferrara, E., De Meo, P., Fiumara, G., and Baumgartner, R. (2014). Web data extraction, applications and techniques. *Know.-Based Syst.*, 70(C):301–323.

Ghani, R., Probst, K., Liu, Y., Krema, M., and Fano, A. (2006). Text mining for product attribute extraction. *SIGKDD Explor. Newsl.*, 8(1):41–48.

Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., and Zhu, X. (2014). Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 601–612, New York, NY, USA. ACM.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goyal, A., Gupta, V., and Kumar, M. (2018). Recent Named Entity Recognition and Classification techniques: A systematic review. *Computer Science Review*, 29:21–43.

Gridach, M. (2017). Character-level neural network for biomedical named entity recognition. *Journal of Biomedical Informatics*, 70:85 – 91.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Komninos, A. and Manandhar, S. (2016). Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California. Association for Computational Linguistics.

Köpcke, H., Thor, A., and Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1-2):484–493.

Kovelamudi, S., Ramalingam, S., Sood, A., and Varma, V. (2011). Domain independent model for product attribute extraction from user reviews using Wikipedia. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1408–1412, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Kuechler, B. and Petter, S. (2004). Design Science Research in Information Systems. (March):1–66.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the*

2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

More, A. (2016). Attribute extraction from product titles in ecommerce. *arXiv preprint arXiv:1608.04670*.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticæ Investigationes*, 30(1):3–26.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pradhan, S., Moschitti, A., Xue, N., Ng, H. T., Björkelund, A., Uryupina, O., Zhang, Y., and Zhong, Z. (2013). Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Putthividhya, D. and Hu, J. (2011). Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Raju, S., Pingali, P., and Varma, V. (2009). An unsupervised approach to product attribute extraction. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 796–800, Berlin, Heidelberg. Springer-Verlag.

Rei, M. (2017). Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130.

Reimers, N. and Gurevych, I. (2017). Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Shen, Y., Yun, H., Lipton, Z., Kronrod, Y., and Anandkumar, A. (2017). Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Shinzato, K. and Sekine, S. (2013). Unsupervised extraction of attributes and their values from product description. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1339–1347, Nagoya, Japan. Asian Federation of Natural Language Processing.

Statista (2019). eCommerce - worldwide: Statista Market Forecast. (Accessed: 4- Dez- 2019).

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Uzuner, Ö., South, B. R., Shen, S., and DuVall, S. L. (2011). 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Wallach, H. M. (2004). Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22.

Wu, F. and Weld, D. S. (2007). Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 41–50, New York, NY, USA. ACM.

Yadav, V., Sharp, R., and Bethard, S. (2018). Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172, New Orleans, Louisiana. Association for Computational Linguistics.

Zhai, Z., Nguyen, D. Q., and Verspoor, K. (2018). Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 38–43, Brussels, Belgium. Association for Computational Linguistics.

Zhang, X. and LeCun, Y. (2015). Text understanding from scratch. *CoRR*, abs/1502.01710.

Zheng, G., Mukherjee, S., Dong, X. L., and Li, F. (2018). Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1049–1058, New York, NY, USA. ACM.