UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Pedro Paulo Valadares Brum

REPRESENTAÇÕES VETORIAIS PARA DESCRIÇÕES DE ITENS EM TAREFAS NÃO SUPERVISIONADAS

Belo Horizonte
2021

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Pedro Paulo Valadares Brum

EMBEDDED REPRESENTATIONS FOR ITEM DESCRIPTIONS IN UNSUPERVISED TASKS

Belo Horizonte
2021

Pedro Paulo Valadares Brum

REPRESENTAÇÕES VETORIAIS PARA DESCRIÇÕES DE ITENS EM TAREFAS NÃO SUPERVISIONADAS

Versão final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Gisele Lobo Pappa

Coorientador: Anisio Mendes Lacerda

Belo Horizonte
2021

Pedro Paulo Valadares Brum

EMBEDDED REPRESENTATIONS FOR ITEM DESCRIPTIONS IN UNSUPERVISED TASKS

Final version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Gisele Lobo Pappa

Co-advisor: Anisio Mendes Lacerda

Belo Horizonte
2021

© 2021, Pedro Paulo Valadares Brum.
Todos os direitos reservados

Brum, Pedro Paulo Valadares.

B893e Embedded representations for item descriptions in unsupervised tasks [manuscrito] / Pedro Paulo Valadares Brum — 2021.
xvi, 95 f. il.

Orientadora: Gisele Lobo Pappa.
Coorientador: Anisio Mendes Lacerda.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f.87-95

1. Computação – Teses. 2. Representação documentária – Teses. 3. Agrupamento de texto – Teses. 4. Processamento da linguagem natural (Computação) - Teses. I. Pappa, Gisele Lobo II. Lacerda, Anisio Mendes. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV.Título.

CDU 519.6*82 (043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Embedded Representations for Item Descriptions in Unsupervised Tasks

PEDRO PAULO VALADARES BRUM

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. GISELE LOBO PAPPA - Orientadora
Departamento de Ciência da Computação - UFMG

PROF. ANISIO MENDES LACERDA - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. RODRYGO LUIS TEODORO SANTOS
Departamento de Ciência da Computação - UFMG

PROFA. SOLANGE OLIVEIRA REZENDE
Instituto de Ciências Matemáticas e de Computação - USP

Belo Horizonte, 14 de Setembro de 2021.

Agradecimentos

Primeiramente, gostaria de agredecer a minha família, em especial aos meus pais, por todo apoio e incentivo que me deram nos estudos. Também agradeço aos meus amigos pelo carinho e pelos momentos de descontração e a alegria. Tudo que vivi com vocês durante esse período foi fundamental para eu superar as dificuldades. Serei eternamente grato.

A minha orientadora, Gisele, agradeço pelo apoio e paciência e pelas oportunidades que me ofereceu durante o mestrado e a graduação. Muito obrigado por ter acreditado na minha capacidade e por todo apoio diante das dificuldades encontradas ao longo do desenvolvimento do trabalho.

Resumo

A maioria dos algoritmos de aprendizado de máquina exige como entrada um vetor de tamanho fixo. Isso torna a área de representação de texto uma área desafiadora de pesquisa em Processamento de Linguagem Natural (NLP), e seus resultados são altamente dependentes da aplicação em questão. Para tarefas de NLP, esse vetor de tamanho fixo geralmente representa uma frase ou um parágrafo. No entanto, construir representações de sentença capazes de capturar as informações semânticas e específicas de um contexto não é uma tarefa fácil. Neste trabalho propomos uma metodologia para resolver um problema real: a identificação de objetos únicos de licitação em bases de dados do Ministério Público Federal de Minas Gerais. Esse cenário traz desafios que vão além dos comumente conhecidos na área de representação de texto, uma vez que queremos agrupar descrições de produtos ou serviços. Essas descrições no geral não seguem a estrutura gramatical de uma sentença na língua portuguesa, já que são formadas em sua maioria por substantivos, adjetivos, e quantidades, essas últimas descrevendo a quantidade de itens comprada/contratada ou a unidade de medida que descreve o item. Dentro do arcabouço proposto, damos ênfase ao problema de representação de texto para algoritmos não-supervisionados. Propomos uma estratégia simples de extração de informações para melhorar a qualidade dos vetores de sentenças, com foco em termos específicos como números e substantivos, e apresentamos uma modificação do Sentence-BERT, que pode ser usada de forma não-supervisionada para geração de embeddings que carregam informações semânticas e sintáticas das descrições. Também identificamos termos numéricos e unidades de medida como os dois componentes principais neste contexto, e mostramos que um método simples de padronização de números tem um efeito significativo nos resultados. Resultados experimentais mostram ganhos do arcabouço proposto em relação a métodos estado-da-arte.

Palavras-chave: Representação de texto, Agrupamento de texto, Vetores de palavras.

Abstract

Most machine learning algorithms require a fixed-size vector as input. This makes the area of text representation a challenging one in Natural Language Processing (NLP) tasks, and its results are highly dependent on the target application. For NLP tasks, this fixed-size vector usually represents a sentence or a paragraph. However, building text representations capable of capturing semantic and context-specific information is not a simple task. In this work, we propose a methodology to solve a real-world problem: the identification of unique objects from public procurement stored in the databases of the Federal Public Ministry of Minas Gerais. These scenarios pose challenges that go beyond those commonly known in the text representation area, as we want to group descriptions of products or services. These descriptions in general do not follow the grammatical structure of a sentence in the Portuguese language, as they are mostly formed by nouns, adjectives, and quantities, the latter describing the quantity of items purchased/contracted or the unit of measure that describes the item. Within the proposed framework, we emphasize the text representation problem for unsupervised algorithms. We propose a simple information extraction strategy to improve the quality of sentence vectors, focusing on specific terms such as numbers and nouns, and present a modification of the BERT siamese network, which can be used in an unsupervised way to generate embeddings that carry semantic and syntactic information from descriptions. We also identify numerical terms and measurement units as the two main components in this context, and show that a simple method of standardizing numbers has a significant effect on the results. Experimental results show improvements from the proposed framework in relation to state-of-the-art methods.

Palavras-chave: Text representation, Text clustering, Word embeddings.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

The rising of the Web and social media has increased the volume of text produced by people dramatically. Texts are present in a variety of contexts, such as social networks (e.g. Twitter, Facebook, Linkedin), short messages services (SMS), chat messages (eg. WhatsApp, Telegram), product reviews and descriptions (e.g. Amazon, Walmart, MercadoLivre), news (e.g. CNN, BBC), electronic health records (EHR), among others. Given the large volume of text produced worldwide and the difficulty of understanding complex texts, there is a need to build automated processes to extract and analyze useful information from them in several contexts.

Because of that, the use of Natural Language Processing (NLP) tasks, such as text classification [Radford et al., 2018], text disambiguation [Li et al., 2013], named entity recognition (NER) [Peters et al., 2018], part-of-speech tagging (POS-tagging) [Owoputi et al., 2013] and question answering (QA) [Devlin et al., 2019; Peters et al., 2018] is growing in many areas. One of the most essential and challenging tasks in this area, regardless of the target task, is to choose a suitable representation for text [Conneau et al., 2017]. For example, if we want to perform the task of text clustering it would be more appropriate to adopt a vector representation instead of the textual representation itself, since there is a distance relationship between scalar and vector values that can be exploited by clustering algorithms. On this matter, there is a number of approaches that can build vector representation for texts, in particular for sentences and paragraphs, such as bag-of-words, tf-idf and embeddings [Kiros et al., 2015; Logeswaran and Lee, 2018; Le and Mikolov, 2014].

Embedded representations are, in particular, mostly produced by methods based on deep artificial neural networks. Deep neural networks (DNNs) are methods inspired by the structure and functioning of the brain, and are able to learn feature hierarchies through multiple stacked layers of neurons [Schifano et al., 2018; Rumelhart et al.,

1986]. The hierarchy of concepts makes it possible for DNNs to learn complicated relationships within the data.

In the context of NLP, DNNs have been revolutionizing the field, as they can build models that are able to learn robust and meaningful representations for text, capturing semantic and syntax relationships between words [Mikolov et al., 2013a; Bojanowski et al., 2017; Pennington et al., 2014a]. These models can be used in an unsupervised manner to build representation for sequences of tokens [Pennington et al., 2014a] or in a supervised way through fine-tuning [Reimers and Gurevych, 2019]. Since pre-trained models and representations can be saved for future usage, the research of NLP tasks is constantly growing.

DNNs can achieve state-of-art results in several NLP tasks through a number of architectures, such as Convolutional Neural Networks (CNN) [Wang et al., 2016], Hierarchical Attention Networks (HAN) [Yang et al., 2016], Long short-term memory (LSTM) [Conneau et al., 2017] and Bidirectional Transformers (BERT) [Devlin et al., 2019]. Although there is a number of different deep neural architectures that can be used to build vector representation for sentences, it is not clear which of these models is more suitable for domain-specific contexts. Regardless of the architecture chosen, most studies in the NLP research field focus on supervised tasks, including text classification [Radford et al., 2018], named entity recognition [Peters et al., 2018], and question answering [Devlin et al., 2019; Peters et al., 2018]. However, there are few works that leverage the quality of different sentence representation on unsupervised tasks, such as text clustering.

Clustering is one of the most popular data mining techniques to deal with unlabeled data and has many applications, such as identifying meaningful patterns and visualization. This task has been used for solving real-word problems, such as detection of cyber-anomalies and policy violations [Sarker et al., 2020a,b]. In short, the idea behind text clustering is to group similar text together based on their meaning.

This dissertation is motivated by a text clustering task to solve a real-world problem, namely the detection of overpriced products and services (also called items or objects) in public procurement. The main difficulty of the problem comes from the fact that only the item descriptions and their prices are available. In addition, product descriptions usually do not follow the same grammatical structures of sentences and have a lot of scalar information, which can represent the number of objects required or a particular specification for the object (e.g., “100 folhas de papel A4”). More precisely, the used dataset extracted from the Public Ministry of the stare includes 196,747 public procurements (e.g., bidding) held between 2015 and 2018 in the state of Minas Gerais, Brazil. The data embraces all areas of public administration and

objects from a variety of areas. Hence, the collection of item descriptions comprises a large variety of words and do not follow any standardized format.

In recent years, the task of generating representation for sentences has been explored as a supervised task, in which deep neural networks are trained on large corpora to later derive sentence embeddings in an unsupervised manner [Reimers and Gurevych, 2019; Conneau et al., 2017]. Most of the methods that explore this strategy are based on a siamese network trained on common semantic textual similarity tasks. Since it is difficult to evaluate the quality of representations on low-level tasks like text clustering, sentence similarity also comes as an alternative way to evaluate the quality of sentence representations obtained by different methods proposed in the literature, as well as the novel approach proposed in this work.

1.1 Motivation

This work is motivated by the problem of building representations for text, particularly for sentences and paragraphs, in tasks where the sentences do not follow a traditional grammatical structure. Despite the growth of deep neural networks (DNNs) to solve similar problems, we are far from reaching a consensus on how to build text representation for unsupervised tasks [Reimers and Gurevych, 2019; Allahyari et al., 2017].

The most commonly used strategy for building embedded representations for sentences is to average word embeddings, which yields rather ineffective results [Pennington et al., 2014b]. Trying to improve the quality of sentence representations, researchers have started to use DNNs to learn sentence representations directly. BERT [Devlin et al., 2019], in particular, set new state-of-the-art performance on various sentence classification and sentence-pair regression tasks, such as semantic textual similarity (STS). Another common strategy to generate sentence embeddings is to derive fixed-size vectors using pre-trained models in an unsupervised way [Reimers and Gurevych, 2019; Conneau et al., 2016]. These models are commonly called encoders and have been extensively explored in recent years. Two main questions need to be answered to build an encoder: what is the best neural network architecture for the target task, and how and on which task should this network be trained for. Most approaches learn sentence encoding in an unsupervised manner, like SkipThought [Kiros et al., 2015] and FastSent [Hill et al., 2016]. However, more recently researchers have been investigating how supervised learning can be used instead.

One of the proposed supervised methods for building sentence representations is Sentence-BERT (SBERT), a modification of the pre-trained BERT network that uses

a siamese network structure to derive semantically meaningful sentence embeddings. This model can be fine-tuned on context-specific data and used to map sentences to a vector space in an unsupervised manner, which can be used by clustering algorithms and other machine learning models. In its seminal paper, SBERT explored natural language inference (NLI) data for training the encoder. In this study, we are also interested in building vector representation for sequences of tokens. However, as previously mentioned, we focus on product descriptions, which have many singular properties and offer more technical challenges in the context of text representation. In addition, conventional language models, such as word2vec [Mikolov et al., 2013a] and GloVe [Pennington et al., 2014a], do not capture efficiently numeric information in text, which are widely present in item descriptions.

As far as we are concerned, this is the first study that focus on improving text representation using semantic and syntax information for supervised and unsupervised tasks. The lack of research in unsupervised NLP tasks, particularly on text clustering, highlights the importance of this work.

1.2 Objectives

The main goal of this work is to investigate an compare methods currently used for building sentence representation, explore new strategies based on semantic and syntactic features for improving the quality of sentence representations, and develop a framework for clustering similar items together based on their descriptions. Following the most popular works on text representation, this study focuses on sentence representation models trained on supervised data. In contrast to most NLP studies, this dissertation focuses on using sentence representations in a unsupervised manner and discusses how these can be applied to solve text clustering, more specifically, in the context of item descriptions. In addition, for investigating the currently used text representation algorithms, we designed and conducted extensive experiments to validate and compare the performance of different models and their respective architectures. The general objective of this work can be narrowed down into three specific goals, driven by the following research questions:

Research Question 1 (RQ1): Which are the most appropriate strategies for generating sentence vectors representations for unsupervised tasks, such as text clustering?

We investigate both simple strategies and more sophisticated methods to generate sentence representations in an unsupervised manner for text clustering. We also leverage the quality of these representations on common semantic textual similarity

tasks, which are used to train most of the models presented in this work. Among the discussed approaches for building sentence vectors are bag-of-words, word embeddings averaging, SIF (Smooth Inverse Frequency) [Arora et al., 2017], InferSent [Conneau et al., 2017], and SBERT [Reimers and Gurevych, 2019].

Research Question 2 (RQ2): Can we enhance language models with semantic and syntax information, such as part-of-speech tags and named entities?

This question arises from the idea that standard models do not adequately address a general linguistic fact, that is, different text components serve diverse roles in the meaning of a sentence. In general, the subject, predicate, and object serve the most important roles of a sentence, as they represent the primary meaning of it. However, this does not apply to item descriptions, the main context of this study. They usually do not follow a grammatical structure as sentences and paragraphs. Nevertheless, item descriptions often contain numbers and units of measure which are very important, as they carry information about the scalar magnitude of objects. Hence, we propose the usage of a supervised sentence representation method that explores the numeric components of item descriptions. We also exploit the usage of these models in other datasets. The main idea is to take advantage of the main components of a specific context to enhance the quality of sentence representations.

Research Question 3 (RQ3): Can we model real-world problems and achieve state-of-the-art results in text clustering in the context of item descriptions?

We propose and evaluate methods more suitable to deal with item descriptions, as they have singular properties in comparison to other sequences of words. Item descriptions usually do not follow the same grammatical structure of sentences and contain information about the scalar magnitudes of objects, providing more technical challenges to build robust sentence representations. We focus on the fact that conventional language models, like word2vec [Mikolov et al., 2013a] and fastText [Bojanowski et al., 2017], do not capture this information effectively.

1.3 Main Contributions

This dissertation presents contributions to the fields of text representation and text clustering, according to the research questions stated in Section 1.2. They are listed below.

A General framework for text clustering and text representation (RQ1): This framework can be used to solve problems that involves text clustering. In this

study, in particular, we group similar items based on their descriptions, solving a real-word problem related to the detection of overpricing in public procurement. As the sentence representations used for these tasks were derived in an unsupervised manner, they can also be used to solve other NLP tasks.

Strategies for improving sentence representations based on exploiting semantic and syntactic information (RQ2): We propose a simple method based on arithmetic operations and component focusing to improve sentence representations using semantic and syntax information. The proposed method is trained on supervised data and can be later used to derive sentence embeddings in an unsupervised manner.

Evaluation of sentence representation models in text clustering (RQ3): The clustering framework was evaluated together with nine different sentence representation models.

A strategy for building synthetic datasets of item descriptions (RQ4): In order to train and evaluate sentence representations obtained by different methods on the context of item descriptions, synthetic datasets were designed. These dataset were created with focus on numbers and units of measure, as they play a crucial role in the meaning of an object description.

Comprehensive experimental evaluation: The numerous approaches that have been presented in the literature motivated us to preform an extensive and comprehensive experimental evaluation, in particular for common semantic textual similarity tasks, which use classification and regression objective functions.

1.4 Text Organization

This dissertation is organized as follows. Chapter 2 presents works from the literature that are related to our research, such as deep neural networks, sentence representation methods and text clustering. Chapter 3 describes the proposed methodology for solving the problem of clustering item descriptions in a real-word dataset. Chapter 4 describes our experimental methodology for sentence representation methods, presenting the datasets characteristics, the experimental setup and the results of the currently used approaches. Next, Chapter 5 presents the new method proposed in this dissertation and discusses our results on semantic textual similarity tasks. Chapter 6 presents and discusses the final clustering results. Finally, Chapter 7 draws conclusions and points out directions of future work.

Chapter 2

Background and Related Work

This study comprises several methods to generate text representations for words and sentences. It involves building language models with a large corpora as well as evaluating the representations in sentence similarity tasks and using them for text clustering. For this reason, this chapter presents previous work on the four main sides of our research: neural architectures (Section 2.1), word embeddings representations (Section 2.2.1), sentence embeddings representations (Section 2.2.2), and text clustering (Section 2.4).

2.1 Neural Architectures

This section presents and discusses a few neural network architectures. We clarify that deep learning architectures have many theories, concepts and details and the focus of our work is to compare the performance of existing algorithms, rather than explaining each architecture in detail. For a more in-depth understanding of the methods, the reader is encouraged to read the references cited in this work. Specifically, we focus on four types of architectures: convolutional neural networks (CNNs), recurrent neural networks (RNNs), attention models, and transformers.

But before going into detail into these architectures, we briefly review a Multilayer perceptron, which is the simplest example of a deep learning model. Neural networks are formed by many single units called artificial neurons. Particularly, each neuron receives inputs, weights them on the basis of their importance, computes a weighted sum of the inputs and finally, adds up an additional input b called bias to properly tune its output value. The output of a neuron is $n_i = \sigma(w_i \cdot x + b_i)$, where w_i and b_i are the weights and bias of the linear transformation and σ is a nonlinear activation function. The nonlinear activation function enables the neural network to

model complex non-linearities of the underlying relations between the inputs and the target variable [Schifano et al., 2018]. As shown in Figure 2.1, a MLP is composed of an input layer, an output layer that makes a prediction about the input and, in between those two, an arbitrary number of hidden layers.

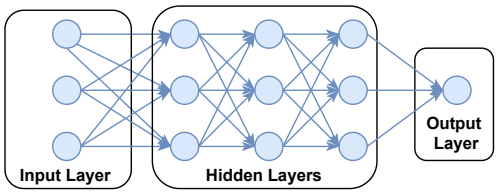


Figure 2.1: Multilayer Perceptron - MLP.

During the training of such models, the weights are usually initialized to small random values and the bias to zero. After that, each input is passed from layer to the next until it reaches the output layer. In this phase, the network computes the output and the error corresponding to the input. The error is calculated on a training or validation set, and then propagated backward changing the values of weights and bias in order to reduce the error itself [Schifano et al., 2018].

An input layer is determined by the number of features in the dataset. For textual data, it can be words or characters. The output layer for classification problems can have n neurons for n-way classification and utilize a softmax function to output the probability of an example to belong to each class. There are other hyperparameters that should be chosen - via cross-validation or in a trial-and-error setting. Hyperparameters are numerical presets that have their values defined prior to the start of the learning process. It is very difficult to determine their optimal values. Most hyperparameters optimization algorithms depend on searching a generic range of values and these are imposed blindly on all sequences [Dong et al., 2018]. Following, we describe some important hyperparameters that can be tuned in the training process of deep learning architectures.:

- Learning rate: controls how much to change the model in response to the estimated error each time the model weights are updated.

- Optimization algorithm: is used to learn the weights of the network. Specifically, it is the process that minimize the error and adjusts the network weights. The most used algorithms are based on gradient descents, each one applying different types of improvement, such as Adagrad, Adam and RMSProp Graves [2013].

- Dropout regularization probability: dropout is a regularization technique for reducing overfitting and improving the generalization of deep neural networks. It is implemented by randomly selecting nodes to be dropped-out from the network, with a given probability, at each weight update cycle.
- Weight initialization: In most cases, weights are initialized randomly. In some finely-tuned settings, weights are initialized using a pre-trained model strategy.

These same hyperparameters are present in CNNs, RNNs and all other neural network architectures. More complex architectures have many different hyperparameters and they are very sensitive to even small changes to them. There are many possible values, which brings a lot of complexity to the process of tuning or choosing the parameters.

2.1.1 Convolutional Neural Networks (CNN)

CNNs were initially created to recognize shapes and patterns in images and audio [Lee et al., 2009; Masci et al., 2013], but later started to be used to train language models and other NLP tasks [Mikolov et al., 2013b; Zhang et al., 2015; Conneau et al., 2016]. Convolutions were first applied to natural language processing tasks by Collobert et al. [2011] in semantic role labelling and later by Kim [2014] in sentiment and question classification. When applied to text instead of images, we have a 1-dimensional array representing the text. Figure 2.2 illustrate how a CNN works with word-based inputs. Each word is represented by a vectorized encoded representation, that might be one-hot encoded or even a real-valued representation.

The network in Figure 2.2 has three different layers that define a CNN: an input layer, a convolution layer and a pooling layer. The convolution layer is where most of the computational operations are. Convolution is a linear operation that involves the multiplication of a set of weights with the input. As illustrated in Figure 2.3, the convolution operation begins at the top of the input matrix. Then the values of the input matrix are multiplied by the corresponding values in the convolution filter (or kernel). All of the multiplied values are added together resulting in a single scalar, which is placed in the convoluted feature vector. After that, the kernel moves down by the length of x pixels, where x is called stride (a parameter of the CNN structure).

This process is repeated until it convolutes the whole input matrix. This architecture allows the network to concentrate on low-level features in the first hidden layer, before assembling them into higher-level features in the next hidden layer, and so on [Conneau et al., 2016; Zhang et al., 2015]. The process of convolution over text can

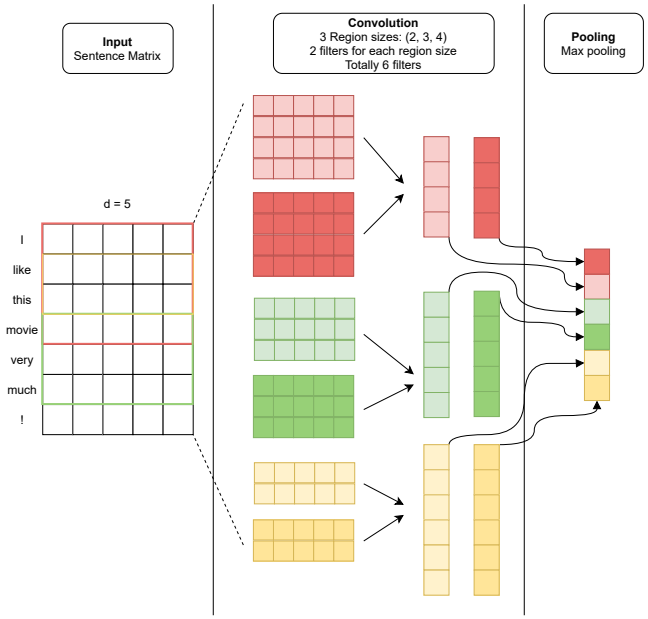


Figure 2.2: Convolutional Neural Network for NLP.

capture k-grams from the sequence of m vectors [Goldberg, 2017]. CNNs might have many layers, as each layer gets distinct feature maps. Its most important hyperparameters are number of kernels and their dimensions, number of convolutions, and types of pooling.

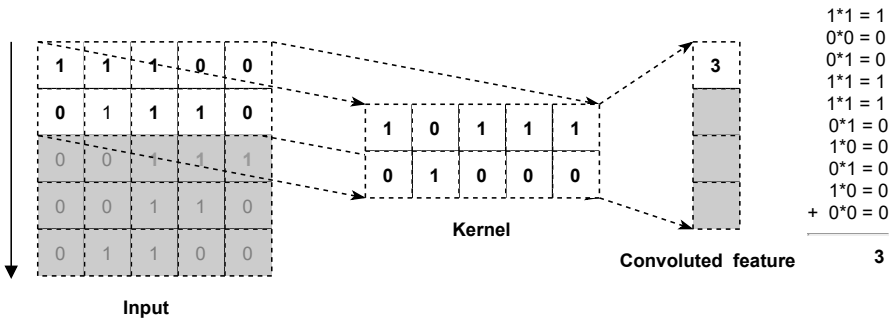


Figure 2.3: Convolution operation.

Figure 2.4 exemplifies the process of pooling over a sentence, performed by the last layer of Figure 2.2. The pooling layer reduces the size of the resulting feature map through an aggregation operation. A pooling neuron does not have weights. It aggregates the inputs using an aggregation function, such as *max* or *mean*. Specifically, the pooling layer summarized the features learned in the previous layers, which helps to prevent overfitting. The advantage of the pooling layer is its ability to inject location

invariance into the network, which means that features can be detected by the network wherever they are on the input.

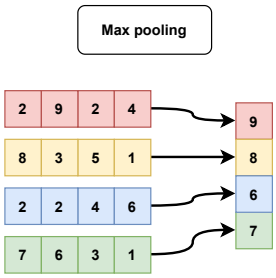


Figure 2.4: Pooling operation.

2.1.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) were created for solving problems where the sequence is more important than the individual items themselves. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation [Cho et al., 2014], speech recognition [Rao et al., 2017], and image captioning [Mao et al., 2014]. As RNNs were designed for sequential data, they are used in many natural language processing applications [Conneau et al., 2017; Rao and Spasojevic, 2016]. Sequential data is basically ordered data in which related inputs follow each other, such as DNA sequence and time series.

Figure 2.5 shows a recurrent neural network. Each RNN neuron/unit takes two inputs at each time step: the input x_t and a hidden state h_{t-1}. In the case of natural language processing tasks the input is one word from the input sentence. The word, however, is represented by a vector. With the input and the hidden state, the RNN unit creates the output y_t and a hidden state h_t, which is used in next time step. A loop allows information to be passed from one step of the network to the next.

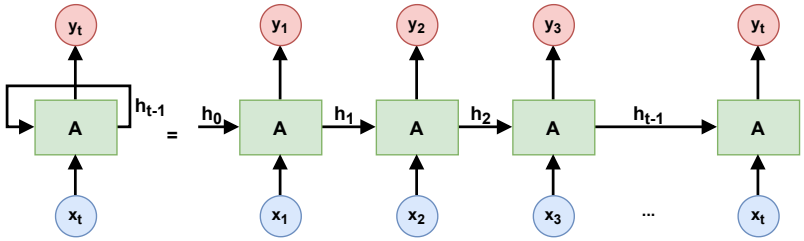


Figure 2.5: Recurrent neural network in compact and expanded representations.

RNNs have many variants. In this study, we will focus on Long short-term memory networks (LSTM) [Hochreiter and Schmidhuber, 1997; Conneau et al., 2017]. LSTMs are a special kind of RNN capable of learning long-term dependencies. They are known to work very well on a large variety of problems [Conneau et al., 2017]. In standard RNNs, the repeating module has a very simple structure, like a single tanh layer. LSTMs also have this chain like-structure, but the repeating module is composed by many neural network layers.

RNNs encouraged the development of the sequence-to-sequence (or encoder-decoder) models. These models are a relatively recent architecture that has opened many possibilities for machine translation, speech recognition and text summarization. Both the encoder and the decoder tend to be recurrent neural networks. An encoder-decoder maps an input sequence to an output sequence, which can be of a different length [Salvaris et al., 2018]. In these models, the last hidden state produced by the encoder is also called context, which is used as the first hidden state by the decoder.

2.1.3 Attention and Transformers

In encoder-decoder models, when the sentence becomes large, the information from its beginning might not be learned well, as the network struggles to retain all available information. Attention mechanisms were introduced in order to solve this problem [Vaswani et al., 2017], and they allow the model to focus on the relevant parts of the input sequence.

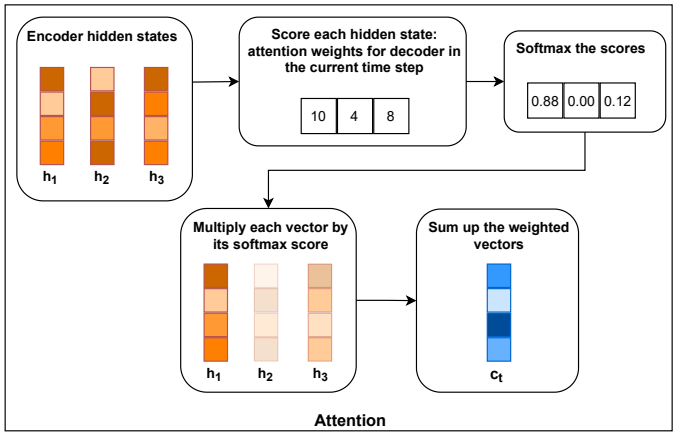


Figure 2.6: Attention at time step t.

Attention models were first introduced in NLP for machine translation tasks [Bahdanau et al., 2014; Luong et al., 2015]. Figures 2.6 and 2.7 illustrate how attention works in a sequence-to-sequence model in a high level. Attention models can also be

used as a tool for interpreting the behavior of neural networks. The first stage of the attention is the encoding, where a set of RNNs receive text as input data and process it to become basically a vector of numbers (hidden states). After that, the encoder passes the hidden states to the decoder, which receives a current word y_t and a hidden state (initially h_{init}). Then the RNN processes its inputs, producing an output and a new hidden state vector (h_t). Once the RNN outputs h_t, it uses the hidden states from the encoder plus h_t to calculate a context vector c_t, which represents the scores from the words around the current word, for the current time step. They are both concatenated into one vector, passing through a feedforward neural network. The output of this network is the word (y_{t+1}) for this time step. This process is repeated for each time step (next words). In that way, the encoder passes all the hidden states to the decoder.

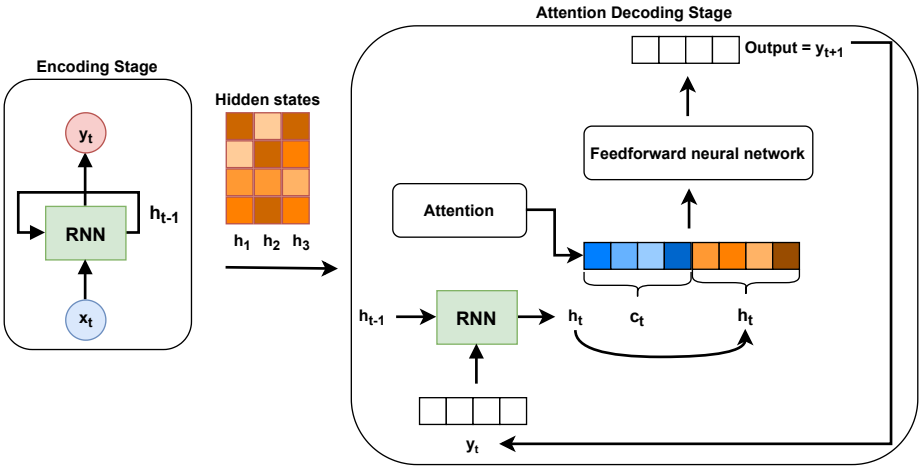


Figure 2.7: Sequence-to-sequence model with attention.

The use of attention has increased the computational complexity for training these networks, as they need to compute a separate context vector for every step of the decoder. Transformers were proposed to mitigate this problem [Vaswani et al., 2017], being an evolution of attention models that do not rely on recurrent units. As shown in Figure 2.8, a transformer has stacked layers of encoders and decoders. The encoder is composed by a self-attention model and a feed forward neural network and the decoder has 3 layers: a self-attention, a encoder-decoder attention and a feed forward layer. Self-attention is defined as an attention mechanism relating different position of a single sequence in order to compute a representation of the sequence. In short, self-attention allows the model to look at the other words in the input sequence to get a better understanding of a certain word in the sequence. The transformer computes self-attention multiple times, thus they name this process as Multi-head Attention.

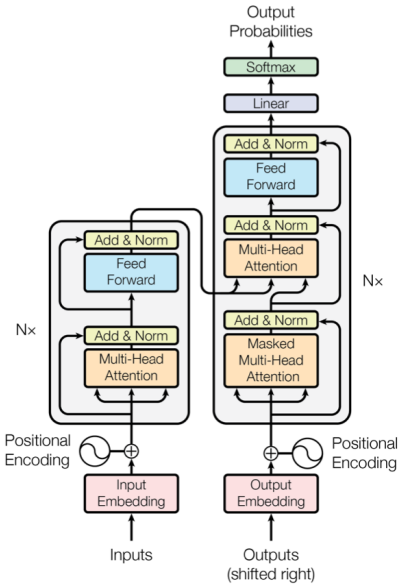


Figure 2.8: Transformer architecture [Vaswani et al., 2017].

Transformers have demonstrated to be a huge improvement over current DNN architectures and the most recent technique applied to NLP tasks. They have overcome many problems but they also have limitations, such as the fixed-length of text input, which leads to context fragmentation. This means that important sentences might be split without considering the semantics.

2.2 Text Representation

This section discusses relevant related work that generates text representations by training language models with large corpora. Mainly, we highlight the word and sentence embeddings methods and discuss briefly how each one of them works. As the focus of our work is on building sentence embeddings representation, we do not explain the word embeddings algorithms in depth, and only give an overview of word2vec [Mikolov et al., 2013a], fastText [Bojanowski et al., 2017] and GloVe [Pennington et al., 2014a]. On the other hand, for sentence embeddings representations, we focus on showing simple strategies like bag-of-words (BoW) [Salton et al., 1975] and tf-idf [Beel et al., 2016] and more robust methods, such as sent2vec [Pagliardini et al., 2018], SIF [Arora et al., 2017], InferSent [Conneau et al., 2017] and SentenceBERT [Reimers and Gurevych, 2019].

2.2.1 Word Embeddings Representations

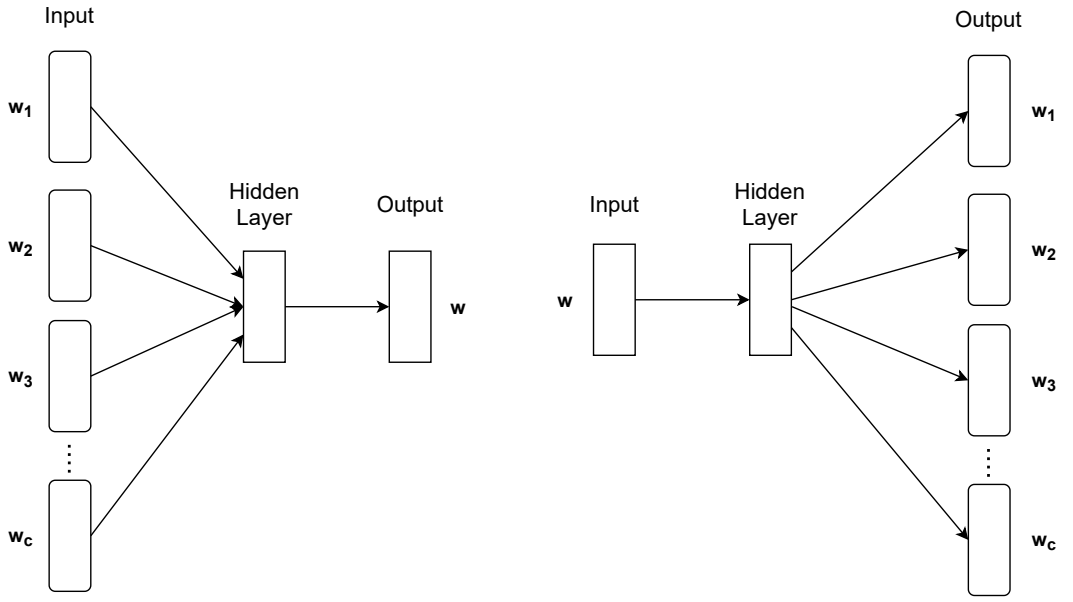
Text representation is an area of research in NLP that has always been broadly studied, and is currently benefitting from the use of neural networks to generate these representations [Devlin et al., 2019; Pennington et al., 2014b; Mikolov et al., 2013b; Melamud et al., 2016]. There are many methods for generating word vectors [Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006]. By using these algorithms on a large corpus of words, we can capture relationships between words, such as gender, verb tense, or country-capital relationships, which have not been possible in early years. Many frameworks for creating word embeddings are currently available as pre-trained vectors to be incorporated into deep learning models in a fully automatic way. Therefore, the use of pre-trained *embeddings* is an essential part of NLP algorithms nowadays, as they have allowed significant performance improvements in several NLP tasks when compared to simpler *embeddings* [Turian et al., 2010].

One of the pioneer works in NLP, well-known as word2vec, was proposed in 2013 [Mikolov et al., 2013a]. Word2vec is a two-layer neural network that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus, also called word embeddings. Mikolov et al. propose the continuous bag-of-words (CBOW), which uses context to predict a target word, and skip-gram, which uses a word to predict a target context [Mikolov et al., 2013b]. CBOW and skip-gram are models created to efficiently construct high-quality distributed vector representations.

As shown in Figure 2.9a, the objective of CBOW is to infer a missing word in a given context of size C. The context is made of a set of words, except the target missing word. As the order of words in the context is not relevant, the model is named “bag of words". The input of the model are C one-hot encoding vectors of size V (the vocabulary size), one for each word in the context (w_1, w_2, w_3, ..., w_C). The desired output is a one-hot encoding vector of size V representing the target missing word, with the actual output being a multinomial distribution achieved by a sofmax regression function.

The objective of skip-gram is to learn the opposite task of CBOW: given a word, infer its surrounding context. The skip-gram architecture is illustrated in Figure 2.9b. The input of the model is a word w (one-hot encoding of size V, the vocabulary size) and the output is a set of words (each word one-hot encoded) within a context window of size C (w_1, w_2, w_3, ..., w_C).

Another type of word embedding that emerged along with word2vec is a count-base model, called Global Vectors (GloVe) [Pennington et al., 2014b]. In summary,



(a) CBOW (b) Skip-gram

Figure 2.9: Neural network architectures.

GloVe generates word embeddings by aggregating the global word-word co-occurrence matrix from a corpus. It is important to note that although GloVe obtains the best results when compared to other state-of-the-art methods, the computational cost of training GloVe word vectors is much higher, specially because it includes the generation of the full co-occurrence matrix for the corpus.

Both word2vec and GloVe ignore the morphology of words, by assigning a distinct vector to each word of the corpus. This is a limitation for language with large vocabulary and many rare words. To solve this problem, Bojanowski et al. [2017] proposed a new method based on the skip-gram model, where each word is represented as a bag of characters n-grams. A vector representation is associated to each character n-gram and the words are represented by the sum of its n-grams. When $n = 3$, the word *where* is represented by the character n-grams <wh, whe, her, ere, re> and the special sequence <where>. The authors also propose a different scoring function to take into account the internal structure of words and training optimizations, such as hierarchical softmax and hashing n-gram features, to significantly accelerate the training process.

Peters et al. [2018] proposed a new way to generate more robust and representative embeddings, named ELMo (Embeddings from Language Models). It extracts features using a left-to-right model and a right-to-left model. The representation of each word is equivalent to the concatenation of left-to-right and right-to-left representations. By using ELMo for specific NLP tasks in text, Peters et al. [2018] were able to obtain state-

of-the-art results for several benchmarks, including: Question Answering, sentiment analysis, and named entity recognition. Melamud et al. [2016] also proposed to generate representations with the task of predicting a word from its left and right contexts using LSTMs. Analogous to ELMo, the model is based on features and is not directly bidirectional.

A break-through in this was a new language model called BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2019], capable of generating representations through left and right context using unlabeled data. BERT involves two steps: pre-training and fine-tuning. In the pre-training step, the model is trained from unlabeled data in two NLP tasks. In the fine-tuning step, the model is initialized with the parameters resulting from the pre-training, and then all parameters are optimized using labeled data in various NLP tasks. BERT obtains state-of-the-art results for various NLP tasks, such as text inference and question answering. In this work, we study and compare the performance of different word representations on common semantic textual similarity tasks and text clustering.

2.2.2 Sentence Embeddings Representations

In addition to word vectors, there are other denser text representations, such as sentence vectors [Kiros et al., 2015; Logeswaran and Lee, 2018] and paragraph vectors [Le and Mikolov, 2014]. While there seems to be a consensus concerning the usefulness of word embeddings and how to use them, this is not clear for representations that carry the meaning of a full sentence [Conneau et al., 2017]. Nowadays, the study of how to derive semantically meaningful sentence embeddings is still a relevant open research question [Conneau et al., 2017; Reimers and Gurevych, 2019].

Table 2.1: Research papers on sentence embedded representations ordered by year.

Method	Paper	Year	Architecture	Input Level	Embeddings
tf-idf	Salton et al.	1975	-	words	-
Average	Pennington et al.	2014	-	words	GloVe
Weighted average (SIF)	Arora et al.	2017	MLP	words	word2vec
Sent2vec	Pagliardini et al.	2018	MLP	sentence	-
InferSent	Conneau et al.	2017	BiLSTM	sentence	GloVe
SentenceBERT	Reimers and Gurevych	2019	BERT	sentence	-

Table 2.1 summarizes important information about a few sentence embeddings methods, including the year, type of architecture for DNN-based methods, input of the method (words, chars or sentences) and if the input of DNN-based methods uses pre-trained word embeddings. The classic text representation model is bag-of-words in which each sentence is represented as a binary vector, considering the presence or

absence of word in the sentence. Another popular representation model is the TF-IDF (*Term frequency - Inverse document frequency*), which considers the frequency of terms in the documents and the rarity of terms in the corpus [Salton et al., 1975]. Although there are many methods for generating sentence vectors, TF-IDF is still a widely used text representation model due to its simplicity [Beel et al., 2016]. It is known that this representation is widely used when the data correspond to a set of documents [Yogatama et al., 2015]. In this case, the sentence is represented by a vocabulary[1] size vector, where all elements are equal to zero except the ones associated with the words that occur in the sentence, which are equal to the TF-IDF values of the terms: TF $\times$ IDF. *Term frequence* (TF) represents the importance of a term t in a document d, and is defined as:

$$tf(t,d) = \frac{\#\text{times the term t appears in the document d}}{\#\text{terms in document d}} \tag{2.1}$$

Inverse document frequency (idf) of a term t represents the weight of the term considering all documents in the corpus, and is defined as:

$$idf(t) = \log\left(\frac{\#\text{documents}}{\#\text{documents that have the term t}}\right) \tag{2.2}$$

Despite being a simple representation, TF-IDF is very sparse and does not add context. It is also important to highlight that this representation could be insufficient for solving a NLP task if all words have low frequency. To mitigate these issues, many methods have been proposed for generating sentence vectors. Most approaches for sentence representation learning are unsupervised, mainly because there is not a consensus about the best supervised task for embedding the semantics of a whole sentence [Hill et al., 2016]. Several works have proposed to use labelled datasets of paraphrase pairs to obtain sentence embeddings in a supervised manner. Another challenge of learning vectors for sentences using supervised tasks is the fact that neural networks can capture the biases of the task on which they are trained, and forget the overall information of the input data. In contrast, learning models on unsupervised tasks makes it harder for the model to specialize [Conneau et al., 2017].

The works on generating sentence embeddings comprise models that compose word embeddings [Le and Mikolov, 2014; Arora et al., 2017] and models based on complex neural network architectures [Conneau et al., 2017; Reimers and Gurevych, 2019]. Kiros et al. [2015] proposed a method called SkipThought, which uses an objective function that adapts the skip-gram model for words to the sentence level, and

[1]Vocabulary: single words in a *Corpus* (text set).

trains an encoder-decoder model to predict the surrounding sentences. Hill et al. [2016] proposed a method called FastSent, which is a sentence-level log-linear bag-of-words model. Like SkipThought, FastSent uses adjacent sentences as the prediction target and is trained in an unsupervised fashion.

While methods like SkiptThought and FastSent require ordered text, where the next sentence is a logical continuation of the previous one, methods like sent2vec [Pagliardini et al., 2018] rely only on an unordered collection of sentences. Sent2vec is a simple unsupervised model that allows to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training the composition and the embedding vectors themselves. The sent2vec model is depicted in Figure 2.10 (adaptation of the sent2vec figure from Agibetov et al. [2018]). Conceptually, the model can be interpreted as a natural extension of the word-contexts from CBOW [Mikolov et al., 2013a] to a larger sentence context. Another way to think about sent2vec is as an unsupervised version of the fastText classification model [Joulin et al., 2017], where the entire sentence is the context and the possible class labels are all vocabulary words. More specifically, the sentence embedding is defined as the average of the word embeddings of its constituent words and the n-gram embeddings present in the sentence. In this way, sent2vec has training and inference complexity as low as simple averaging methods. When using sent2vec, the sentence embedding v_s for S is calculated as:

$$v_s = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w \tag{2.3}$$

where $R(S)$ is the list of n-grams (including unigrams) present in sentence S.

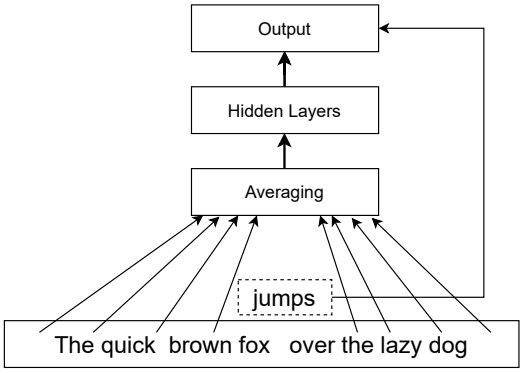


Figure 2.10: Sent2vec model: the context corresponds to the entire sentence.

It has been shown in the literature that the use of naive methods, such as word embeddings averaging, to obtain sentence embeddings can outperform more sophisticated algorithms [Wieting et al., 2016; Pennington et al., 2014b; Arora et al., 2017].

Pennington et al. [2014b] proposed the use of plain averaging of unigrams vectors to obtain sentence embeddings. On the other hand, Arora et al. [2017] proposed a model where the sentences are represented as a weighted average of pre-trained word vectors, followed by a post-processing step of subtracting the principal component.

There are multiple methods to encode sentences using neural networks, such as InterSent [Conneau et al., 2016], Sentece-BERT [Reimers and Gurevych, 2019] and convolutional neural networks [Kim, 2014; Kalchbrenner et al., 2014; Hu et al., 2014]. As previously described, CNNs use convolutional filters to capture local dependencies in terms of context windows and apply a pooling layer to extract global features. InferSent uses labeled data of the Stanford Natural Language Inference dataset [Bowman et al., 2015] and the NLI datasets [Williams et al., 2018] to train a siamese BiLSTM network with max-pooling over the output. Conneau et al. [2017] showed that InferSent outperforms unsupervised methods like SkipThought, and found that NLI datasets are suitable for training sentence embeddings. As illustrated in Figure 2.11, the architecture of InferSent uses a shared sentence encoder (siamese network) that outputs a representation for two sentences A and B. Once the vectors are generated, 3 matching methods are applied to extract relations between sentence representations u and v: (i) concatenation of the two representations $(u,\ v)$; (ii) element-wise product $u * v$; and (iii) absolute element-wise difference $|u - v|$. The resulting vector is fed into a classifier consisting of a linear and a softmax layer.

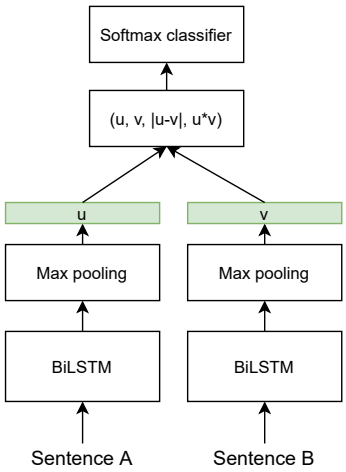


Figure 2.11: InferSent architecture: BiLSTM with max pooling. The two BiLSTM networks have tied weights (siamese network structure).

Other works discuss the generation of sentence embeddings through fine-tuning methods. Sentence encoders and sentences that generate text representations can be pre-trained from unlabeled text and optimized for supervised tasks [Dai and Le, 2015;

Howard and Ruder, 2018; Radford et al., 2018]. One of the advantages of this approach is that few parameters need to be optimized. Reimers and Gurevych [2019] developed a method called Sentence-BERT (SBERT) for generating sentence vectors. Like Conneau et al. [2017], Reimers and Gurevych [2019] use a siamese network structure and the NLI datasets (including SNLI and Multi-Genre NLI)) for training sentence embeddings. But instead of using a BiLSTM, they use BERT with 3 different structures and objective functions: (i) classification objective function, (ii) regression objective function, and (iii) triplet objective function. The objective function depends on the available training data. Another difference between InferSent and SBERT is the matching methods applied to the resultant vectors u and v. SBERT only considers two methods: concatenation of the two representations (u, v) and absolute element-wise difference $|u-v|$. SBERT embeddings were particularly evaluated using a similarity measure like cosine-similarity and Manhattan/Euclidean distance, and common semantic similarity (STS) tasks. In that way, Reimers and Gurevych [2019] were able to show that SBERT outperforms other state-of-the-art sentence embeddings methods.

Our study comprehends a comparative analysis of the aforementioned sentence embeddings methods as well as the generation of vectors for product descriptions (particularly of public procurement data). To our knowledge, this work is the first attempt to exploit a dataset composed by product descriptions for building generic sentence encoders using supervised and unsupervised methods.

2.3 Text Representation Enhancement

Several works have proposed enhancements for word and sentence representations using simple strategies, such as multimodal models [Rahman et al., 2020] and component focusing [Yin et al., 2020]. Multimodal models are typically applied when the input data is not only text but also nonverbal data (visual and acoustic). These models usually generate a shift to the text representation based on its visual and acoustic modalities.

Rahman et al. [2020], for instance, proposed an attachment to BERT called Multimodal Adaptation Gate (MAG), which allows BERT to receive multimodal nonverbal data as input, as shown in Figure 2.12. This component receives as input the lexical vector (word embedding) as well as its visual (V_i) and acoustic (A_i) accompaniments. The vectors are combined through an attention gate into a new vector that is added (shifting) to the input lexical vector to produce the final representation. In our study, we follow a similar idea. But, instead of using acoustic and visual inputs, we use text

information about the lexical input, such as part of speech and named entities.

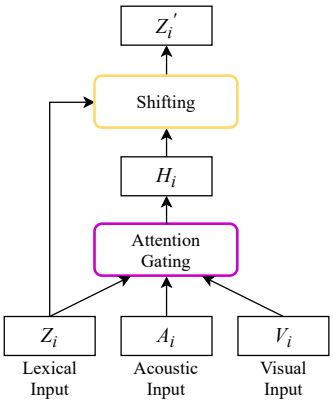


Figure 2.12: Multimodal Adaptation Gate (MAG), which takes as input a lexical input vector with visual and acoustic correspondents.

Also exploring other aspects of text, Yin et al. [2020] proposed a modification of Sentence-BERT which uses component focusing (CF-BERT). Figure 2.13 shows the overall architecture of the proposed sentence representation model. In that manner, the sentence representation is obtained by the basic sentence part S_{basic}, and a component-enhanced sentence part S_{cf}. The basic sentence part S_{basic} contains the complete sentence information, whereas the component-enhanced part S_{cf} contains the crucial sentence information. More precisely, CF-BERT uses dependency parsing to obtain S_{cf}, keeping only the subject, predicate and object of the sentence. As Reimers and Gurevych [2019], Yin et al. [2020] also trained the sentence representation model using the NLI datasets and used common sentence similarity benchmarks to evaluate the final sentence vectors.

We used the sentence representation model proposed by Yin et al. [2020] with a few modifications. Following a similar idea, our study also considers other text information to build the component-enhancement part, such as named entities and part-of-speech tags, and other datasets to train and evaluate the model. As one of our goals is to generate vector representations for product descriptions, we focus on numerical terms and units of measure. In the context of text representation enhancement, we also consider the method of canonizing numbers proposed by Zhang et al. [2020]. They showed that a simple algorithm of canonizing numbers from the input data can have a great effect on the results. In view of its crucial relevance to our study, Chapter 5 gives a more detailed explanation of this proposed sentence representation model.

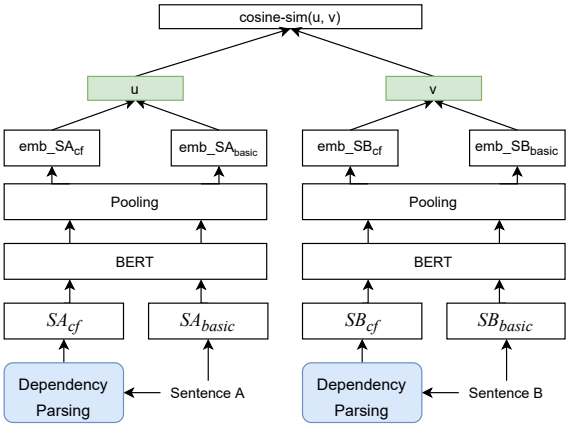


Figure 2.13: The overall architecture of the component focusing bidirectional encoder representations from transformers (CF-BERT) model.

2.4 Text Clustering

Sentence vectors obtained by different text representation methods can be used in several natural language processing tasks, supervised and unsupervised [Allahyari et al., 2017]. Unsupervised learning methods comprehend techniques that try to find hidden structure out of unlabeled data. In the context of text data, the most popular data mining algorithms are clustering and topic modeling. Therefore, a common method to address text clustering and semantic search is to map each sentence to a vector space such that semantically similar sentences can be clustered (grouped) together without manual effort. In this way, a collection of sentences can be segmented into partitions, where sentences in the same cluster are more similar to each other than those in other groups.

Text clustering embraces a wide range of applications, such as in classification [Bekkerman et al., 2001], visualization [Cadez et al., 2003], and sentence organization [Zeng et al., 2004]. Clustering algorithms can be applied to different levels of text granularities, which can be paragraphs, sentences, terms (tokens) or descriptions.

Naive methods to obtain vector representation for sentences, like bag-of-words and TF-IDF, can achieve poor results in comparison to more sophisticated methods [Radu et al., 2020]. There are three main aspects that need to be considered when generating sentence vectors for text clustering:

1. Dimensionalty: as text data is sparse, the bag-of-words and TF-IDF representations can have a very large dimensionality. In other words, even though the sentence may have only a few words, its vector can be very large due to the size of the vocabulary.

2. Context: words of the vocabulary of a given collection of sentences are commonly correlated with each other. Thus, it is necessary to use algorithms which take word correlations into consideration.

3. Normalization: since the number of terms in a sentence can vary, normalizing the sentences vectors before the clustering process is crucial.

In principal, any clustering method can be used in the context of text data, including k-means, x-means [Pelleg and Moore, 2000], DBSCAN [Ester et al., 1996a], HDBSCAN [Campello et al., 2013]. They have varied trade-offs in terms of effectiveness and efficiency. In our study, we focus on three clustering algorithms: x-means [Pelleg and Moore, 2000], HDBSCAN [Campello et al., 2013] and SBERT [Reimers and Gurevych, 2019].

K-means clustering is one of the partitioning algorithms (or flat-centroid based) which is widely used in data mining. This method partitions a collection of sentences into k clusters, finding k centroids used to assign a cluster to each one of the sentences based on similarity. K-means suffers from three main problems: the number of clusters is supplied by the user; the single partitioning of the data enables the assignment of noise to clusters; and the implicit assumption that clusters have Gaussian distributions. In an attempt to solve the first problem Pelleg and Moore [2000] proposed X-means, which does not require a pre-defined number of clusters k.

In contrast, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [Ester et al., 1996a] is a density-based clustering algorithm capable of finding noise within the data. However, being a density-based approach, DBSCAN suffers from the difficulty of parameter selection. Trying to solve this problem, Campello et al. [2013] propose HDBSCAN, the hierarchical version of DBSCAN. HDBSCAN generates a complete density-based clustering hierarchy from where a simplified hierarchy composed only of the most significant clusters can be easily extracted.

Chapter 3

Identifying Unique Objects from Brazilian Public Procurements

The main objective of this dissertation is to solve a real-world problem, which is the identification of unique items and an estimate of their price within a set of descriptions of items previously purchased in public procurements. This chapter describes the problem and the methodology proposed to solve it.

When a public procurement for purchase or contracting is opened, the institution must establish a reference value (price) to guide competitors in setting the prices of their products. The reference price has several purposes, such as establishing support for the expense budgeting process, substantiating the criteria for acceptability of proposals, substantiating the economicity of the purchase or contract, among others. Besides, reference prices can be used as parameters for identifying overpricing in future public procurements.

The data we work with consists of a collection of items collected from the system of the Public Ministry of the State of Minas Gerais, and is written in Portuguese. The system currently contains a set of 196,747 public procurement held between 2015 and 2018. These public procurements embrace all areas of public administration and objects of the most varied natures. In addition, they can be part of different modalities, such as invitation, electronic auction, and face-to-face auction. Particularly, we focused on the items whose descriptions have at least one numeric term or unit of measure, as one of the main focus of our study relies on these components. The items collected are quite diverse, not restricted to any type of items.

Furthermore, text clustering can be used to group similar items together based on their descriptions, and reference prices for specific products or services can be defined according to the cost of the objects that compose the groups. Therefore, this value

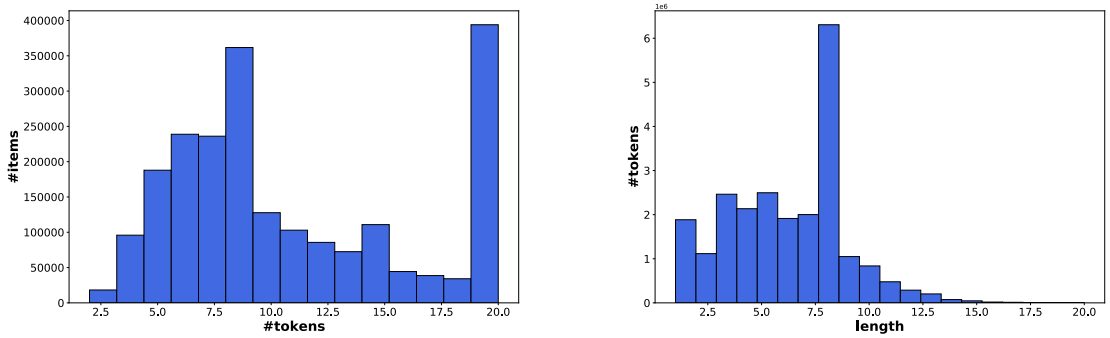
can then be used as a parameter for analyzing and diagnosing possible overpricing in public procurement.

This chapter characterizes the item descriptions and introduces the methodology proposed to solve the problem, which details its four major phases: (i) data cleaning, (ii) information extraction, (iii) text representation and (iv) grouping.

3.1 Characterization of Item Descriptions

The dataset we work with comprises 2,149,533 items, where 2,096,664 are unique (i.e., with a unique description according to an exact match). From the collection of descriptions, we identified 156,311 unique tokens. Since a large number of items have repeated words, we removed the duplicate tokens from all of them. For example, the description "LIXA FERRO 36 LIXA FERRO 36" has 3 unique tokens, "lixa", "ferro" and "36", resulting in the description "lixa ferro 36".

Figure 3.1a shows the histogram for description length. We can observe that most descriptions are formed by up to 15 tokens (1,587,769 descriptions - 73.87%) and a significant amount has 20 tokens (364,715 descriptions - 16.97%). The descriptions have an average size of 10.85 and the the smallest descriptions have 2 tokens. It is important to mention that this characterization considers the descriptions after data cleaning, as detailed in Section 3.2.



(a) Item descriptions length. (b) Tokens length

Figure 3.1: Histograms.

Figure 3.1b shows the histogram for token length, i.e., number of characters a token has. Observe that a considerable number of tokens has size equal to 1 (1,884,646 tokens - 8.08%). Furthermore, 91.58% of the tokens (21,366,891) have up to 10 characters.

Figure 3.2 shows the word cloud for the 2,149,533 item descriptions present in the dataset. The size of the words is proportional to their frequency. For a better visualization, all numeric terms were removed, as they have high occurrence in the dataset. We can observe that words that indicate the characteristics of objects are recurrent: "cor", "branco", "plastico", "papel", "caixa", "tipo", "largura" and "tamanho". It is interesting to note that words indicating quantity (units of measure) also occur very often. Some of them are: "ml", "cm", "unidade", "caixa" and "embalagem". This demonstrates the importance of considering numbers to differentiate items, since an object can have in its description "100 mg" or "20000 mg", which are very different.

Figure 3.2: Word cloud of item descriptions. The size of words is proportional to the their frequency in the dataset, and colors do not encode information.

Figure 3.3 presents the word cloud of the most representative bigrams of the item descriptions. Again, the size of bigrams is proportional to their frequency. Some of the most frequent bigrams are: "aco inoxidavel", "pacote com", "cor branco", "prestacao servico", "para uso", "mg comprimido", "embalagem com" and "tamanho cm". Note that one of the most frequent bigrams is "prestacao servico", which means that an item can also refer to a service provision, such as equipment leasing, maintenance, cleaning, security, gardening, and repairs.

3.2 Proposed Methodology

For grouping similar objects we propose a methodology that consists of four major phases, shown in Figure 3.4: (i) data cleaning, (ii) information extraction. (iii) text representation and (iv) grouping. Each of these phases comprises a set of operations,

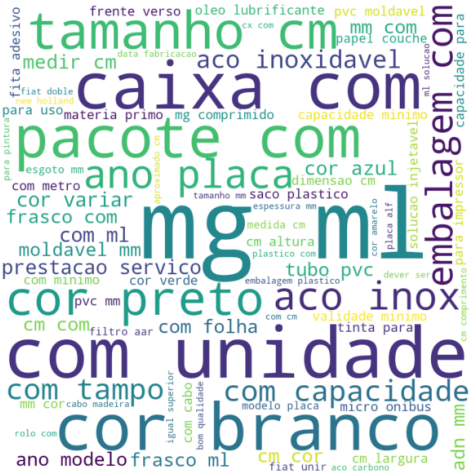


Figure 3.3: Word cloud of the most representative bigrams in the item descriptions. The size of words are proportional to the their frequency in the dataset, and colors do not encode information.

where dotted boxes correspond to optional steps. The next sections present in detail each of the steps of the proposed framework.

3.2.1 Text Cleaning

The first step of the proposed methodology is text cleaning, which involves four basic macro-operations: (i) preprocessing, (ii) tokenization, (iii) spellchecking, and (iv) lemmatization. The main objective of this phase is to improve the quality of the input dataset. As a preprocessing step, all descriptions were lower-cased and stop words, accents, punctuation and tokens with more than 20 characters were eliminated. In addition, repeated words were also removed, since a lot of items have repeated terms in the beginning of their descriptions. In summary, six operations are applied in the item descriptions in the preprocessing step:

1. Lower case: all characters are lower cased.
2. Stop-words removal: meaningless words are removed. Among them are articles, prepositions and conjunctions.
3. Punctuation removal: all punctuation marks are removed from descriptions.
4. Non-alphanumeric characters removal: characters like '#', '', '&', '$' are removed from the descriptions.
5. Long terms removal: words that have more than 20 characters are removed.

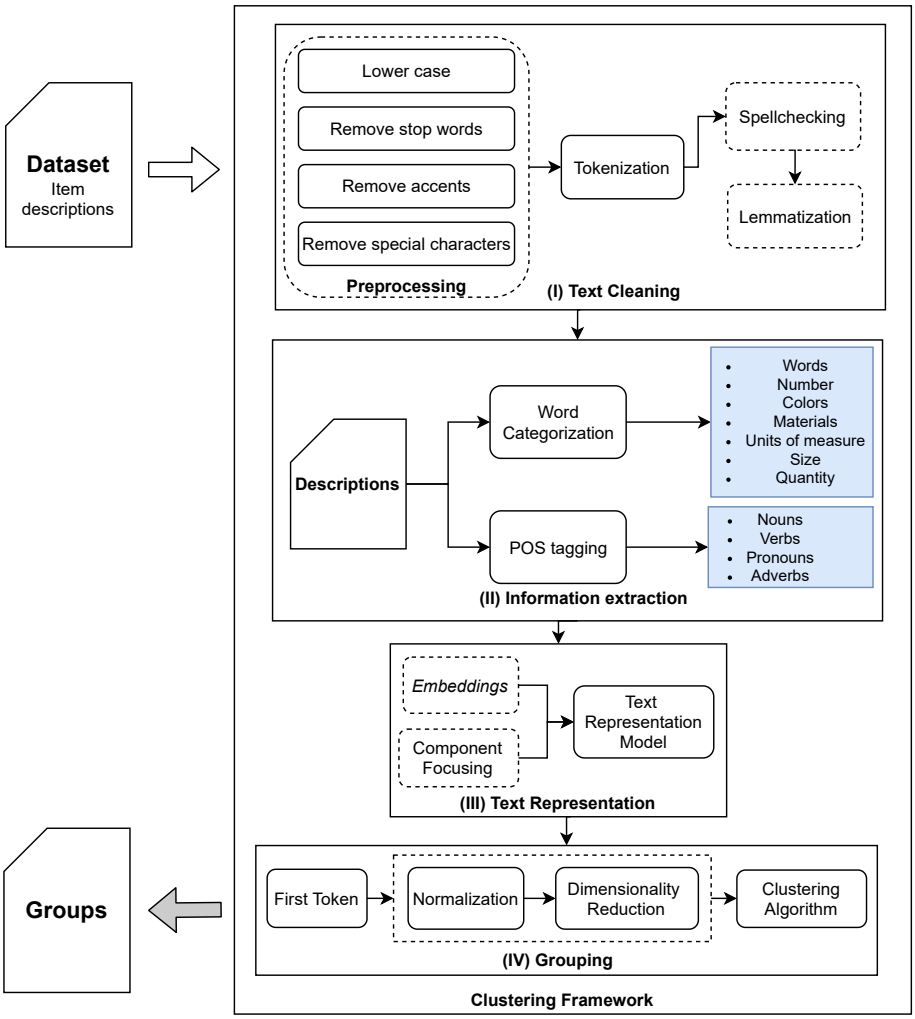


Figure 3.4: Clustering Framework.

6. Canonicalizing numbers: replace all numbers with their representation in scientific notation, a combination of an exponent and a power (for example 314.0 is represented as 3.14e+02).

Some stop words were not removed, as they can play an important role to the meaning of a particular item. This is the case of "para", "com" and "sem", which are helpful to identify item characteristics, such as “agua com gas”, “agua sem gas”, “pano para pia” or “jogo de soletrar nao recomendavel para menores de 12 anos”. Furthermore, it is important to highlight that numbers are very important to describe objects, as they carry information about scalar magnitudes. However, some tokens match a sequence of digits (1 to 9) followed by a sequence of characters, such as "100mg" "200mm", "50cm". Many of these tokens correspond to numbers that occur next to a unit of measure. In this way, we defined as a preprocessing step the inclusion of spaces between numbers

and units of measure.

Table 3.1 presents some examples of item descriptions. The first and second columns show the descriptions before and after text cleaning, respectively. We can see that the descriptions have a lot of noise (repeated words, sequence of digits and meaningless characters, non-alphanumeric characters) and that there is no pattern. For example, the object "005499 Topiramato 100mg" after preprocessing becomes "5499 Topiramato 100mg". From this text it is possible to identify four tokens: "5499", "topiramate", "100" and "mg". In this case, preprocessing was unable to remove the "5499" token from the sentence, since it is interpreted as a word. We highlight that even after text cleaning, some tokens that do not add information to the objects occur in the descriptions.

Table 3.1: Example of item descriptions.

Description	**Description after preprocessing**
COLAGENASE 0,6UI/G	colagenase 0.6 ui g
CAMPO OPERATÓRIO 45CMX50CM	campo operatorio 45 cm x 50
Diclofenaco de Sodio - 75mg /3ml	diclofenaco sodio 75 mg 3 ml
Serviço de Recapagem de Pneus Misto 1100 x 22	servico recapagem pneu misto 1100 x 22
PAPEL SULFITE, OFICIO 9,7G/M2, 216X330MM 500FOLHAS	papel sulfite oficio 9.7 g m2 216 x 330 mm 500 folha
SOLUCAO DE BICARBONATO DE SODIO 8,4% 250ML SIST.FECHADO	solucao bicarbonato sodio 8.4 % 250 ml sist fechado
SULFADIAZINA DE PRATA 120 GRAMAS	sulfadiazina prata 120 grama
CATETER INTRAVENOSO PERIFÉRICO Nº 14, COM SISTEMA DE PROTEÇÃO.	cateter intravenoso periferico n 14 com sistema protecao
CIPROFLOXACINO, 500 MG	ciprofloxacino 500 mg
005499 Topiramato 100mg	5499 topiramato 100 mg

The next step of text cleaning involves tokenization, spell checking, and lemmatization (in that order). Basically, tokenization turns a pre-processed description into a vector of words (i.e., tokens[1]). The lemmatization step consists of replacing each word with its canonical form. Suppose w is a word and c is its canonical form. The lemmatization process is equivalent to applying a function C for each word w in the database such that $C(w) = c$. It is interesting to apply word lemmatization, since most of the words present in the database correspond to variations of canonical words in gender, number and degree. To perform the lemmatization, a dictionary[2] of inflected portuguese words was used. The dictionary has 880,000 inflected words and 9,072,338 inflections. In addition to the word lemma, the dictionary also shows the part-of-speech tag of the words. As the same canonical form can contain different variations, noun variations were given priority.

[1]Token is a part of the text structure and can be a phrase, word, punctuation, etc.

[2]`http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/files/Formato_DELAF_PB.pdf`

Spellchecking corrects misspellings in the words. The corrector used for this task is based on the Levenshtein distance (or editing distance). The Levenshtein distance between two words A and B corresponds to the minimum number of operations needed to transform one word into another (A -> B or B -> A). For this distance, the operations allowed are insertion, removal and replacement of characters. The spellchecking was applied only in words that are not in the dictionary[3]. More precisely, we consider that similar words have a distance of up to 2 operations. Each term that does not occur in the word dictionary is replaced with the most similar word from the corresponding list of words.

3.2.2 Information Extraction

After text cleaning, the proposed framework extracts some relevant information from the set of tokens output. The idea of this step is to identify the category of each term in the item descriptions, which is our case can be categories related to item descriptions or POS tagging. This information can be used later on to build representations for items, as explained in Chapter 5. For this step we considered the following categories, defined after an initial characterization of the dataset:

1. Unit of measurement (Unidades de medida): "litro", "litros", "gramas", "miligrama", "ml", "kg", "v", "cm".

2. Colors (Cores): "preto", "branco", "azul", "vermelho", "claro", "escuro", "roxo".

3. Materials (Materiais): terms that describe different types of materials that compose an item. For example: "plastico", "aco", "metal", "inox", "inoxidavel", "madeira", "pvc", "prata", "ouro".

4. Numbers (Números): all numeric terms. Ex.: "14", "1", "2015", "500", "300", "one", "two", "ten", "II", "IV".

5. Size (Tamanho): "pequeno", "grande", "medio".

6. Quantity (Quantidade): terms that describe the form of presentation of the items and/or their quantities. For example: “pacote”, “pct”, “comprimido”, “cartela”, “unidade”, “und”.

[3]Corpus NILC/São Carlos: `https://www.linguateca.pt/acesso/corpus.php?corpus=SAOCARLOS`.

7. Words (Palavras): any term that does not belong to any of the above categories. For example: "papel", "pneu", "dipirona", "dea", "mascara", "servico", "carne", "banana".

The token categorization was implemented using a dictionary of words. Tokens that do not belong to any specific category are automatically included in the "Words" category. Table 3.2 presents some examples of structured descriptions.

Table 3.2: Item descriptions after the word categorization step.

Description	**Description after word categorization**
cloridrato piridoxina 10 mg	{"palavras": ["cloridrato", "piridoxina"], "unidades_medida": ["mg"], "números": ["10"], "cores": [], "materiais": [], "tamanho": [], "quantidade": [] }
rodo grande 60 cm	{"palavras": ["rodo"], "unidades_medida": ["cm"], "números": ["60"], "cores": [], "materiais": [] "tamanho": [] "quantidade": ["grande"] }
fita adesiva autoclave 19 mm x 30 m	{"palavras": ["fita", "adesiva", "autoclave", "x"], "unidades_medida": ["mm", "m"], "números": ["19", "30"], "cores": [], "materiais": [], "tamanho": [], "quantidade": [] }
infra pedestal dimmer fisioterapia termoterapia lampada 220 v vermelho	{"palavras": ["infra", "pedestal", "dimmer", "fisioterapia", "termoterapia", "lampada"], "unidades_medida": ["v"], "números": ["220"], "cores": ["vermelho"], "materiais": [], "tamanho": [], "quantidade": [] }

In order to validate the predefined word categories, we ask 8 people to manually label words of item descriptions. To perform this validation, we generated four items samples, each containing 150 descriptions. The samples were generated following a distribution of part-of-speech tags and medications. We focused on medication descriptions since they are very frequent in the dataset and often have units of measure.

Moreover, medications often have tokens that can be mislabelled in other descriptions, such as "ferro", "cobre", "zinco", among others. The four samples have the following distribution: 20% of the descriptions containing only nouns; 20% of descriptions containing only medications; 20% of descriptions containing only nouns and medications; 20% of descriptions containing some noun and/or medicine; and 20% of descriptions containing words without a part-of-speech tag.

Furthermore, the samples were created in a stratified way according to the sizes of the descriptions. The following description size ranges were uniformly defined for choosing the set of 150 descriptions: [1,2], [3-6], [7,11], [12, 20] and [20, max(size)]. Each one of the four samples do not have common items and were evaluated by two different people. The evaluation consisted of the inclusion or exclusion of terms in one of the predefined category. After the validation, new words were included in the dictionary of word. In addition to the word categorization, the clustering framework also runs POS-tagging.

3.2.3 Text representation

The next step in the methodology is to build a vector representation for the item descriptions to cluster them afterwards. Note that here we receive the description already categorized in terms of word categories and POS, and the first step here is to select which POS tagging classes and word categories will be used for text representation.

As described in Chapter 2, different types of representations can be used to encode the sentence, including bag-of-words, word embedding, and sentence embeddings. Our focus is on sentence embeddings based on neural networks, and the methods considered in this study were chosen based on two criteria: (i) their popularity in academia; (ii) the easiness of reproducing the implementations as faithfully as possible to the original publication.

We chose a set of five supervised and unsupervised methods for sentence representation, namely SIF, Sent2vev, CNN, InferSent and SBERT. In addition, we also consider the sentence representation model proposed in Chapter 4, a modification of SBERT that focuses on specific components of text to enhance the quality of the output vectors. All methods selected involve the use of a siamese network to train the model using common sentence similarity tasks. This structure is depicted in Figure 3.5.

Having said that, it is important to mention that the most common task used in the literature to evaluate sentence representation methods is sentence similarity. Given a pair of sentences, the method can return two types of information: whether

the sentences are the same (0 or 1) or their level of similarity (a score). In the first case the problem is modeled as a classification task, and in the second as a regression task. The choice of how to model the problem depends on the available training data. As showed later, we have also evaluated the selected methods in other traditional datasets from the literature for sentence similarity considering both a classification and a regression problem.

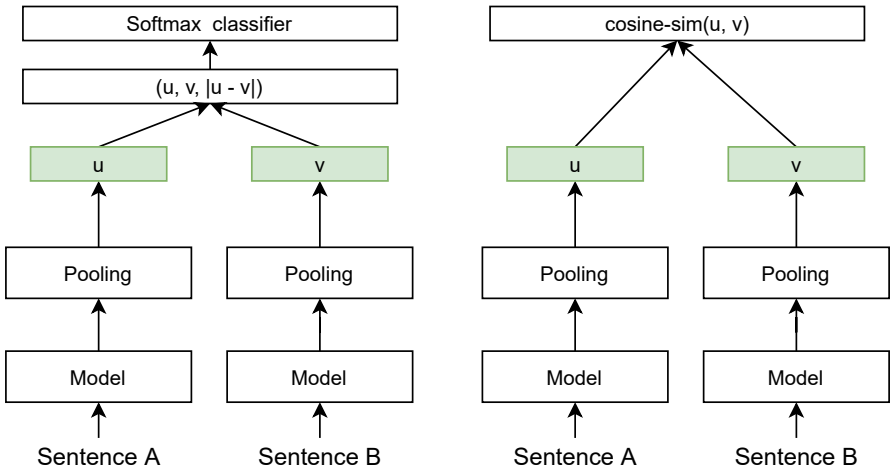


Figure 3.5: Siamese network architecture for classification (left) and regression (right). The two models have tied weights.

Since we want to compare the sentence representation methods, we did not explore in depth the architecture of the models. Hence, for SBERT and InferSent we used the same architecture presented in the original papers and explained in Section 2.2.2. For CNN we implemented a simple architecture that consists of 3 stacked layers, where each one of them has different convolving kernel sizes: 1, 3 and 5. For InferSent, we used 10 epochs, a batch-size of 16, 512 hidden features, Adam optimizer with learning rate 1e-3, a linear learning rate warm-up over 10% of the training data and max pooling. For SBERT, we used 4 epochs, a batch-size of 16, Adam optimizer with learning rate 2e-5, a linear learning rate warm-up over 10% of the training data and mean pooling. All CNN, InferSent and SIF models use the fastText word embeddings trained on the dataset of item descriptions with 300 dimensions. We chose fastText over GloVe and word2vec, as it uses character ngrams information to train the word embeddings.

For classification, we optimize the cross-entropy loss over the output o. The output is calculated by concatenating the sentence embeddings u and v with the element-wise difference $|u - v|$ and multiplying it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = softmax(W_t(u, v, |u - v|)) \tag{3.1}$$

where n is the dimension of sentence embeddings and k the number of labels.

For the regression task, we calculate the cosine similarity between the two sentence embeddings u and v and use the mean squared error loss to optimize the model.

3.2.4 Grouping

After the construction of the representation for item descriptions (using one of the approaches described above), the last step of the proposed approach starts by performing normalization and dimensionality reduction. We observed that these two steps have great impact on the quality of clustering, but they are optional.

A distance function is one of the main components of distance-based clustering algorithms. Normally, the distance between word embeddings is done using the cosine distance instead of the Euclidean distance [Hartmann et al., 2017]. However, not all implementations of clustering algorithms allow the choice of cosine distance. To get around this situation, it is possible to use a strategy in such a way that the result of clustering using Euclidean distance is proportional to that given by the cosine distance. Normalization transforms the item vectors into vectors whose lengths in the vector space are equal to 1. In other words, when using a normalization, the space corresponding to the data is transformed into a spherical format. From the normalization, it can be seen that the cosine distance (1 - cos(x), where x is the angle between the vectors) is directly proportional to the Euclidean distance, since the scalar product of two unit vectors is simply the cosine of the angle between them. This is due to the dot product formula, since the lengths of both vectors are 1.

After the normalization phase, the dimensionality reduction of the vectors is performed to obtain denser vectors. To perform this step, the UMAP (Uniform Manifold Approximation and Projection) is used [McInnes et al., 2018]. Given a set of X-dimensional item vectors, UMAP transforms each vector of this set into a Y-dimensional vector, where $Y < X$. Therefore, using UMAP, it is possible to represent the information contained in X dimensions in a more condensed way. This can also reduce or even remove the noise caused by highly correlated variables and, in this way, the clustering algorithm will have a greater ability to group similar items together. In this study, the item representations were reduced to 15 dimensions, according to a manual analysis. It is important to highlight that the normalization and dimensionality reduction steps are performed for both sparse vectors generated by bag-of-words and for dense vectors generated by word embeddings and sentence representation models.

The final step is the clustering process itself. In datasets such as the one used in this paper, where the variety of items is huge, there are advantages in using simple

heuristics to generate preliminary groups. This initial grouping process is helpful as: (i) due to the large number of items, clustering algorithms can take a long time to run; (ii) some items have similar descriptions, but do not correspond to the same product, which can lead to poor results; (iii) to understand the difficulties of grouping item descriptions, having less chaotic and smaller numbers of groups helps; (iv) it sets a baseline approach for grouping items. These heuristics are described next.

3.2.4.1 Heuristics for pre-grouping item descriptions

In this section, five heuristics for grouping items based on their descriptions are presented. Three methods consider unigrams (one token) and two consider bigrams (two tokens). These heuristics are very simple and do not involve any sophisticated NLP or machine learning techniques. Fundamentally, they group the descriptions according to the exact match between unigrams or bigrams representing the descriptions. The implemented strategies use only the tokens in the category "Words" ("Palavras") of the descriptions. It is important to highlight that these heuristics were first implemented before we had a complete understanding of the problem that we need to solve. They are:

Most Frequent Token A simple way to represent a particular item is by the most important word present in its description. We arbitrarily defined that the most important word of an item description is the most frequent word in the dataset. For example, consider the description "pinca biopsia aco inox". Initially, the object is represented by 4 tokens: "pinca", "biopsia", "aco" and "inox". Suppose the most frequent token in the dataset of this description is "aco". Thus, the most important token of the description is "aco" and the object is represented only by this unigram (term).

Two Most Frequent Tokens Another way to group objects is using the two most frequent words in their descriptions. In this case, similarly to the previous approach, the most important words consist of the two words with the highest frequency in the dataset. For example, consider again the description "pinca biopsia aco inox". Suppose the most frequent token in the dataset is "aco", and the second most frequent is "inox". Thus, the most important tokens of the description are "aco" and "inox" and the object is only represented by the bigram {"aco", "inox"}. If the description has only one word, just the first token is considered.

First Token The third approach consists of selecting the first token of the descriptions. In this case, the most important word in a description is the word in the first position. For example, for the item "pinca biopsia aco inox", the token "pinca" would be selected, different from what happens in the first and second heristics.

First and Second Token Another simple way to represent a given object is by its first and second tokens. In this case, the most important words of a description correspond to the first and second word. For example, for the descriptions "pinca biopsia aco inox", the tokens "pinca" and "biopsia" are selected, different from what happens in the first approach where the tokens "aco" and "inox" are selected.
Most Frequent Bigram A simple way to represent a particular object is by the most important (in this case, frequent) bigram in its description. Again, for the description "pinca biopsia aco inox", it is possible to identify 3 bigrams: {"pinca", "biopsia"}, {"biopsia", "aco"} and {"aco", "inox"}. Suppose the most frequent bigram in the dataset is {"aco", "inox"}. Thus, the item is only represented by this bigram.

We present a comparison of the strategies described above. This comparison involves both quantitative (i.e., statistics regarding the groups obtained) and qualitative (i.e., which descriptions were actually grouped following each strategy) analysis. Tables 3.3 and 3.4 present the quantitative comparisons between the proposed approaches. Analyzing the results shown in these two tables, we highlight the following:

- The *Most Frequent Token* heuristic generated 11,719 groups, and 43.12% of them include a single item description. Its largest group is described by the stopword "com", with 516,846 objects. The *Two Most Frequent Tokens*, in turn, generated 157,130 groups, 50.52% formed by only one item. The groups have an average of 13.7 items and the largest group has 122,333 items.

- The *First Token* heuristic generated 26,648 groups, 46.97% formed by only one item. The groups have an average of 80.64 items and the group sizes present a much smaller variance (equal to 540,657.45) in comparison to the most frequent token. In addition, the largest group is "papel" and comprises 37,486 items. The *First and second token* heuristic generated 227,778 groups, with an average of 9.43 objects per group. Its largest group has 9,580 items.

- In the most frequent bigram heuristic, 52.08% of the 147,028 groups generated have only one item. The mean and the variance of the group sizes are similar to the *two most frequent tokens* heuristic, and its largest group has 15,437 objects.

Table 3.5 presents examples of groups for each grouping strategy. We can observe that frequency-based approaches (i.e., Most Frequent Token, Two Most Frequent Tokens, and Most Frequent Bigram) are able to group well items in small and medium-size groups. The same is not true for larger groups, where it is common to find terms that represent qualifiers and quantifiers, such as "cor" and "tamanho". While these tokens

Table 3.3: Number of groups found by each grouping strategy.

Group Size	**Grouping Strategy**				
	Most Frequent Token	**2 Most Frequent Tokens**	**1º Token**	**Most Frequent Bigram**	**1º and 2º Token**
1	5,053 (43.12%)	79,384 (50.52%)	12,517 (46.97%)	76,574 (52.08%)	121,632 (53.40%)
(1, 5]	3,070 (26.20%)	47,131 (29.99%)	6,762 (25.38%)	42,959 (29.22%)	66,751 (29.31%)
(5, 10]	932 (7.95%)	11,974 (7.62%)	1,962 (7.36%)	10,397 (7.07%)	15,922 (6.99%)
(10, 100]	1,763 (15.04%)	15,988 (10,18%)	3,565 (13.38%)	14,109 (9.60%)	20,149 (8.85%)
(100, 1000]	673 (5.74%)	2,457 (1,56%)	1,429 (5.36%)	2,712 (1.84%)	3,150 (1.38%)
>1000	228 (1.95%)	196 (0,12%)	413 (1.55%)	277 (0.19%)	174 (0.08%)
Total	11,719	157,130	26,648	147,028	227,778

Table 3.4: Statistics for each grouping approach.

Statistics	**Grouping Strategy**				
	Most Frequent Word	**2 Most Frequent Words**	**1º Token**	**Most Frequent Bigram**	**1º and 2º Token**
Mean	183.36	13.67	80.64	14.62	9.43
First quartile	1.00	1.00	1.00	1.00	1.00
Median	2.00	1.00	2.00	1.00	1.00
Third quartile	9.00	4.00	7.00	4.00	3.00
Standard deviation	5,277.42	360.23	735.29	150.78	73.42
Variance	27,851,178.19	129,763.75	540,657.45	22,734.94	5,390.04
Max	516,846	122,333	37,486	15,437	9.580
Min	1	1	1	1	1

are relevant to the descriptions, they are not capable of representing well a collection of items, since they are very generic. On the other hand, grouping strategies based on the first token or on the first and second tokens tend to lead to more domain specific groups. In general, the first token of an item description is the name of the product the description is referring to.

Considering both quantitative and qualitative analysis, we understand that the first token (third heuristic) tends to better represent groups of items. We chose to use this approach to perform an initial grouping of the items for 5 main reasons: (i) it is the approach with the second lowest number of groups formed by only one item; (ii) despite the high variance of group sizes, we believe that it is adequate given the varied collection of descriptions; (iii) high average of items per group; (iv) it obtained a reasonable number of groups (26,648); (v) the obtained groups represent well the collection of items.

However, this heuristic has some drawbacks. We observed that most of the groups that have only one item correspond to objects whose description has spelling errors or two words not separated by space. Another limitation is the fact that items with similar descriptions but referring to different objects are grouped together, such as

Table 3.5: Examples of groups generated by each grouping strategy.

Group Size	Grouping Strategy				
	Most Frequent Word	**2 Most Frequent Words**	**1º Token**	**Most Frequent Bigram**	**1º and 2º Token**
1	'hodrocortizona', 'telescopico', 'citoprofeno'	('oficio', 'pm2'), ('fitomonadiona'), ('basico', 'farinha')	'fitomonadiona', 'flauda', 'iodoforno',	('iogurt', 'light'), ('ficha', 'famila'), ('luva', 'roscalvel'),	('iogurt', 'light'), ('condimento', 'corante'), ('flauda', 'geriatrico'),
(1, 5]	'lavanda', 'feijoada', 'varal',	('diazepam', 'dosagem'), ('ferro', 'largura'), ('eixo', 'tras'),	'amendoa', 'neosaldina', 'fluconozol'	('diazepam', 'dosagem'), ('aco', 'martelo'), ('externar', 'boca'),	('fio', 'triplex'), ('confeccao', 'grelha'), ('vasilha', 'leiteiro')
(5, 10]	'banheira', 'reuquinol', 'capacete',	('diesel', 'locacao'), ('duplo', 'rosca'), ('corante', 'natural'),	'rolha', 'capota', 'fomulario',	('duplo', 'paralelo'), ('aprox', 'caixa'), ('cinarina', 'cpr'),	('diazepam', 'dosagem'), ('jarra', 'temperado'), ('camisa', 'coronel')
(10, 100]	'hipoclorito', 'bandeira', 'cartilha'	('aluminio', 'fio'), ('porta', 'sem'), ('interno', 'uso')	'abafador', 'conversor', 'dedetizacao'	('alcool', 'gasolina'), ('aluminio', 'caneco'), ('lombo', 'largo'),	('equipamento', 'informatica'), ('extensao', 'cabo'), ('microfone', 'mao'),
(100, 1000]	'pilha', 'tesoura', 'coletor'	('ext', 'parafuso'), ('alcalino', 'pilha'), ('cabo', 'item'),	'diazepam', 'feixe', 'azeitona',	('parafuso', 'ext'), ('caminhao', 'mercedes'), ('para', 'rede'),	('veiculo', 'automotor'), ('broca', 'esferico'), ('servico', 'mecanico'),
>1000	'com', 'sem', 'uso'	('com', 'embalagem'), ('com', 'para'), ('para', 'servico')	'parafuso', 'fio', 'alcool'	('acucar', 'cristal'), ('data', 'fabricacao'), ('alcool', 'etilico'),	('alcool', 'etilico'), ('pilha', 'alcalino'), ('soro', 'fisiologico'),

"papel toalha" and "papel higiênico". Although they define different products, they are present in the same group "papel". However, this is something we expect, otherwise the application of a clustering algorithms would be worthless.

3.2.4.2 Clustering pre-grouped items

To refine the groups generated by the heuristics proposed, we use the HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm. It is a hierarchical clustering algorithm that runs DBSCAN [Ester et al., 1996b] for multiple values of *epsilon* (eps) and combines the results to find clusters that offer the best stability. The *epsilon* parameter defines the maximum distance between two instances for one to be considered as in the neighborhood of the other. This relationship goes from instances (in our case, descriptions) that are fully connected (i.e., totally similar) to instances that are totally disconnected by the defined value of epsilon. Running DBSCAN for multiple epsilon values allows HDBSCAN to find groups of varied densities (unlike DBSCAN, which has fixed density). The smaller the epsilon value, the greater the number of divisions performed on the collection of points (instances). It is worth mentioning that HDBSCAN has a module that identifies instances that are considered as outliers (i.e., noise). HDBSCAN is applied for each group generated using the first token (FToken) strategy that contains at least min_size items, where these minimal cluster size is a user-defined parameter.

Chapter 4

Evaluating Models for Sentence Representation

One of the main objectives of this study is to find the most appropriate sentence representation model for semantic textual similarity tasks and text clustering (Research question 1). In this chapter, our main goal is to reproduce the most frequent used sentence representation algorithms in a controlled experimental environment in order to evaluate their contribution to the chosen task. Note that we define a sentence as a text of any size.

We first provide a description of the datasets, including the process followed to generate synthetic datasets where we were able to evaluate the representations. Next, we detail experimental setup and parameter tuning, followed by an overview on how to deal with numeric tokens. Finally, we present the results considering the problem of sentence similarity under a regression and classification perspectives.

4.1 Datasets

We assess the effectiveness of sentence representation models considering two scenarios: a classification and a regression setup. In the literature, the datasets used to evaluate these tasks are different, and we have used the same setup used in previous studies for both tasks. For the experiments where sentence similarity seem as a classification task, we consider four datasets: the NLI dataset [Williams et al., 2018], the Amazon-Walmart dataset [Mudgal et al., 2018], and the Portuguese and English synthetic datasets of item description pairs generated as part of this dissertation. For the regression task, we also consider our two synthetic datasets and two others: the Semantic Textual Similarity benchmark (STSb) and the SICK-Relatedness dataset, as detailed below.

4.1.1 Benchmarks from the Literature

As previously mentioned, Reimers and Gurevych [2019] and Conneau et al. [2016] used the NLI (Natural Language Inference) datasets [Bowman et al., 2015; Williams et al., 2018] to train their proposed models, namely Sentence-BERT and InferSent. As these works confirmed that NLI datasets are suited for training sentence representations models, we also consider them to train and evaluate the performance of the models.

Natural Language Inference (NLI) is the task of deciding if a premise entails the hypothesis, if they are contradictory or if they are neutral. Commonly used NLI datasets are SNLI and MultiNLI. SNLI embrace a collection of 570,000 English sentences pairs manually annotated considering the follow categories: contradiction, entailment, and neutral. Table 4.1 shows one example of sentence pairs for each label. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. Like Reimers and Gurevych [2019], we also combine SNLI and MultiNLI (which we just call NLI), in order to have a larger dataset.

Table 4.1: Sentence pairs of NLI dataset (SNLI + Multi-Genre).

Sentence A (Premise)	**Sentence B (Hypothesis)**	**Label**
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping.	contradiction

The second dataset used is the Amazon-walmart dataset[1], which was originally designed for the entity matching task [Das et al., 2017; Mudgal et al., 2018], which finds data instances that refer to the same real-world entity. More precisely, this dataset comprises 12,694 product-title pairs of two labels: duplicate and non-duplicate. For performance reasons, we undersampled the non-duplicate product-titles pairs, ending up with a dataset composed by 3,462 product titles pairs, where 2,308 (67%) pairs are non-duplicate items.

Table 4.2 describes the class distribution for each dataset considered for the classification objective function. The large scale datasets are balanced and the small ones unbalanced. Table 4.3 summarizes the distribution of words for each dataset.

For the regression setup, as in Reimers and Gurevych [2019], we also evaluate the performance of the sentence embeddings models considering common Semantic Textual Similarity (STS) tasks in a supervised manner. More precisely, we considered the STS benchmark (STSb) [Cer et al., 2017] and the SICK-Relatedness dataset [Marelli et al.,

[1] `https://sites.google.com/site/anhaidgroup/useful-stuff/data`

Table 4.2: Datasets Statistics.

Dataset	Size	# sentences	Vocabulary size	# Classes	Minor class	Major class
NLI	981,382	1,962,764	84,423	3	326,370	327,954
Amazon-walmart	3,433	6,866	6,045	2	1,143	2,290
Synthetic-en	100,000	200,000	17,082	2	30,000	70,000
Synthetic-pt	100,000	200,000	17,045	2	30,000	70,000
Synthetic-en (regression)	100,000	200,000	17,439	1	-	-
Synthetic-pt (regression)	100,000	200,000	17,419	1	-	-
SICK-Relatedness	9,927	19,854	2,460	1	-	-
STS benchmark	8,628	17,256	21,964	1	-	-

Table 4.3: Datasets Vocabulary Statistics.

	Words distribution					
Dataset	Minor	1st Quartile	Median	Mean	3rd Quartile	Major
NLI	0.00	7.00	10.00	12.39	15.00	401
Amazon-walmart	1.00	7.00	10.00	11.42	14.00	82.00
Synthetic-en	3.00	3.00	3.00	3.51	4.00	5.00
Synthetic-pt	3.00	3.00	3.00	3.64	4.00	6.00
Synthetic-en (regression)	3.00	3.00	3.00	3.51	4.00	5.00
Synthetic-pt (regression)	3.00	3.00	3.00	3.65	4.00	6

2014]. These datasets provide labels between 0 and 5 on the semantic relatedness of sentence pairs. The STS benchmark (STSb) is a popular dataset to evaluate supervised semantic textual similarity systems. The data includes 8,628 sentences pairs from three categories: captions, news, and forums, which are divided into training (5,749), validation (1,500) and test (1,379) splits. The SICK-Relatedness dataset contains 9,927 pairs of sentences, where 4,500 are for training, 4,927 for test, and 500 for validation. We provide a description and sample instances of these datasets in Table 4.4.

Table 4.4: Semantic Textual Similarity (STS) tasks description and samples.

Dataset	Task	Sentence A	Sentence B	Output
Sentences Involving Compositional Knowledge Semantic Relatedness (SICK-R)	To measure the degree of semantic relatedness between sentences from 0 (not related) to 5 (related)	A woman with a ponytail is climbing a wall of rock.	The climbing equipment to rescue a man is hanging from a white, vertical rock.	1.8
Semantic Textual Similarity Benchmark (STSb)	To measure the degree of semantic similarity between two sentences from 0 (not similar) to 5 (very similar)	A woman picks up and holds a baby kangaroo.	A woman picks up and holds a baby kangaroo in her arms.	4.6

4.1.2 Synthetic Dataset

As explained before, the main objective of this dissertation is to work with unconventional sentences, which do not necessarily follow the grammatical rules and other

patterns of traditional sentences. Table 4.5 presents examples of item descriptions from our real-world dataset. We can observe that these descriptions do not follow any pattern and often involve numbers and units of measure. We hypothesize that simple methods are unable to generate reasonable representations for these item descriptions.

Table 4.5: Examples of item description from real-world dataset.

Description
COLAGENASE 0,6UI/G
VINAGRE 750 ML.
SACO P/LIXO 30 LT
PARAFUSO 35 X 25MM
Serviço de Recapagem de Pneus Misto 1100 x 22
PAR DE BIELETAS DA SUSPENSAO DIANTEIRA SPACEFOX 2011/2012
COLA BRANCA 90G, 1 QUALIDADE, LAVAVEL
BANDEJA PAPEL DESC. RETANGULAR.
APRAZ 0,5 MG 30CPR
BANANA PRATA

In this context, one of our main objectives is to generate sentence representations that capture information about the scalar magnitudes of objects. Having that in mind, we designed synthetic datasets in English and Portuguese that focus on numerical terms and units of measure, such that the representations generated can carry information about these types of components. For generating the synthetic dataset, we considered pairs of units of measure and the conversion value between them, as well as random items descriptions. In other words, for pairs of the same object we could have a description with different measures but that are equivalent in their conversions.

We considered nine physical quantities, which are summarized in Table 4.7: length, volume, mass, area, time, computational, energy, potency, frequency. It is important to highlight that the selected items were chosen based on their appearance in real-world datasets, such as the amazon-walmart dataset.

Table 4.6: Synthetic datasets description and samples.

Dataset	**Task**	**Description A**	**Description B**	**Output**
Synthetic-en for Classification Objective Function	To predict the label of two descriptions: 0 (non-duplicates) and 1 (duplicates)	electrical wire 25.9 m	electrical wire 27,600 in	0
		gasoline 727 l	gasoline 722,000,000 mm3	1
Synthetic-en for Regression Objective Function	To measure the degree of semantic similarity between two descriptions from 0 (not similar) to 1(very similar)	electrical wire 718 cm	electrical wire 1.17 dam	0.37
		gasoline 11.5 l	gasoline 1,500,000 mm3	0.87

The generated synthetic dataset comprises a collection of 100,000 product descriptions pairs, where each of the 9 physical quantities have the same number of instances (approximately 11%). For each physical quantity, we also defined a proportion of 30%

duplicate and 70% non-duplicate product pairs. To generate a pair of items for a given physical quantity, we considered all possible combinations of units of measure and their conversion values. More precisely, for a given pair of units we considered a value in the range [1, 1000] to obtain the correspondent equivalency. Furthermore, for the duplicate pairs, we increased or decreased the conversion value by 5% with a 50% probability. Therefore, even if the values of a given item pair are slightly different, they can be duplicates. The generation of the synthetic dataset for classification is described in Algorithm 1.

Data: U units of measure pairs, P list of products, Y physical quantity
Input : c class, p probability of noise, x min value, y max value
Output: $\{P_1, P_2\}$ product descriptions pair
$p \leftarrow getUnitsPair(U, Y)$;
$u \leftarrow getConversionValue(p)$;
$d \leftarrow getItem(I, Y)$;
$w_1 \leftarrow random([x, y])$;
if *c is duplicate* **then**
    $P_1 \leftarrow d + w_1 + p_1$;
    $w_2 \leftarrow w_1 * u * random([0.95, 1.05])$;
    $P_2 \leftarrow d + w_2 + p_2$;
else
    $P_1 \leftarrow d + w_1 + p_1$;
    $w_2 \leftarrow random([x, y]) * u < 0.95 * r$ or $random([x, y]) * u > 1.05$;
    $P_2 \leftarrow d + w_2 + p_2$;
end

Algorithm 1: Synthetic Dataset Generation for Classification.

For the regression dataset we also considered the items and units of measure shown in Table 4.7. However, instead of assigning a class (duplicate or non-duplicate) to each product pair, the algorithm assigns a score that represents how similar the two sentences are. Particularly, for each item pair we define a score in the range [0.1, 1.0] such that the higher the score the more similar are the sentences. The generation of the synthetic dataset for regression is described by the Algorithm 2. As the synthetic dataset for classification, this dataset also contains 100,000 product pairs such that each one of the 9 physical quantities have the same number of instances. We provide a description and sample instances of the synthetic datasets in Table 4.6.

4.1.3 Data Preprocessing

All datasets were submitted to the following preprocessing steps:

Data: U units of measure pairs, P list of products, Y physical quantity
Input : x min value, y max value
Output: $\{P_1, P_2\}$ product descriptions pair
$p \leftarrow getUnitsPair(U, Y)$;
$u \leftarrow getConversionValue(p)$;
$d \leftarrow getItem(I, Y)$;
$w_1 \leftarrow random([x, y])$;
$score \leftarrow random([0.1, 1.0])$;
$w_2 \leftarrow random([x, y]) * (1 - score)$ or $random([x, y]) * (2 - score)$;
$P_1 \leftarrow d + w_1 + p_1$;
$P_2 \leftarrow d + w_2 + p_2$;

Algorithm 2: Synthetic Dataset Generation for Regression.

1. Standardization of text to lowercase: this is important to avoid semantic differentiation of similar words based on case.

2. Removal of stop words: words that are not relevant to the meaning of the text, such as articles, prepositions, some verbs among others.

3. Removal of non-alphabetic characters: this procedure is also related to avoiding semantic differentiation of similar words in free natural text (e.g. minimizing differentiation by errors in word accentuation) and exclusion of characters that are not informative to the current application (e.g. punctuation).

4. Removal of words longer than 20 characters: minimizing the chance of occurring "noise" words in text.

5. Canonizing numbers: replace all numbers with their representation in scientific notation, a combination of an exponent and a potency (for example 314.0 is represented as 3.14e+02) -see Section 4.4.

After that, the following operations are also applied to text: tokenization, spell checking and lemmatization transforms a pre-processed description into a vector of words (i.e., tokens). Lemmatization, on the other hand, finds and transforms the words of the descriptions into its base words. Thus, lemmatization further increases the degree of standardization of the database.

Spelling correction corrects errors in the words of the descriptions (missing or excessive letters). The corrector used for this task is based on the Levenshtein distance (or editing distance) between two words. The Levenshtein distance between two words A and B corresponds to the minimum number of operations needed to transform one word into another (A -> B or B -> A). In the definition used for this distance, the operations allowed are insertion, removal and replacement of characters. From the

Table 4.7: Items and units of measure considered for each physical quantity.

Physical Quantity	**Items**	**Units of measure**	**#Units**
Length	flexible cable, electrical wire, craft foam ball, triplex cable, pvc pipe, duplex wire, silk rope, plastic ruler, steel metal ruler, floor squeegee, adhesive tape, foam roller, wall mount fan, cleaning cloth, foam insulating tape, dental floss, thread seal tape, masking tape, metallic matrix tape, tape, barbed wire, black canvas, paper roll, blue paper roll, green paper roll, crepe paper roll, toilet paper roll, garden hose	km hm dam m dm cm mm yd ft in	10
Volume	gasoline, ethanol, diesel, juice, soda, water, disinfectant, bleach, garbage bag, acrylic paint, liquid hand soap, milk, alcohol, chlorine, water tank, oxygen	in3 ft3 gallon l m3 ml mm3 cm3	8
Mass	bean, rice, soy, potato, carrot, banana, apple, grape, pineapple, tomato, corn, cement, brick, carbamazepine, omeprazole, hydrochlorothiazide	kg g mg tonne lb oz	6
Area	card, paper card, cloth, ceramics, bond paper, blue bond paper, red bond paper ,laminated paper, pvc liner, cardboard sheets, tissue paper	cm2 m2 km2 yd2 ft2 in2	6
Time	service, consultancy, construction, contract, cd, dvd	s min hour day week month	6
Computer	notebook, computer, cell phone, hard drive, external hd, video card	bit byte kb mb gb tb	6
Energy	defibrillator, air conditioner	kWh btu cal joule	4
Potency	power supply, fluorescent lamp, reactor, electronic reactor, electric generator, lighting system	hp kw BTU/s joule/s watt	5
Frequency	memory, computer, notebook, processor, engine, antenna, tonner cartridge, fan, motor, ultrasound, microcomputer	hz khz mhz ghz thz	5

spell checker, you can get the list of words that have a distance less than or equal to x for a given word A. The spell checker is applied only to words that do not match in

a word dictionary. For pre-processing, we consider that similar words have a distance of up to 2 operations in relation to the target word. Each term that does not occur in the word dictionary is replaced with the most similar word from the corresponding list of similar words. In this way, it is possible to correct part of the spelling errors that occur in object descriptions.

4.2 Experimental Setup

In our experiment we use the K-fold cross-validation, which consists of randomly splitting the data into K independent folds. At each iteration, one fold is retained as the test set, and the remaining $K-1$ are used as training set. Figure 4.1 gives a practical example: if we split a dataset into 5-folds we obtain 5 different splits, where each split has 80% of the data as training and 20% as testing. The experiments presented in this work were executed using a 5-fold cross-validation procedure.

We focus on sentence representation methods that are based on neural networks. However, since word embeddings averaging and bag-of-words are able to achieve results comparable to state-of-the-art algorithms in some tasks, we also used them as baselines. Concerning methods that use pre-trained word embeddings we considered word vectors generated by three different methods: word2vec, fastText, and GloVe, with vector size equals to 300. For fastText, the size of the context window was defined as 10, with negative sampling (5 negative examples) and default initial learning rate of 0.025.

In the case of bag-of-words and word embeddings averaging, as they are simpler algorithms, we added two dense layers after the pooling layer of the networks presented in Figure 3.5. Particularly, for bag-of-words the pooling layer is omitted, since in this method we only take the frequency of the words in the sentences to build the input vectors.

For BERT, the official Google artificial intelligence (AI) team provides a variety of pre-trained models for different languages and different model sizes. As Reimers and Gurevych [2019], we also experiments SBERT with two setups: only training on specific NLP tasks, e.e.g, NLI, and then on specific NLP tasks.

To evaluate the broader utility of the different sentence representation models, we evaluate their performance for common semantic textual similarity tasks. For classification, all models are compared using micro-averaged F_1 ($MicroF_1$) and macro-averaged F_1 ($MacroF_1$), which are standard information retrieval measures [Yang, 1999]. On the one hand, macro-averaged F_1 corresponds to averaging the F_1 score over all classes, which in turn is the harmonic mean between precision and recall for each class c, as

shown in Equation 4.1:

$$F_1(c) = \frac{2 \times Precision(c) \times Recall(c)}{Precision(c) + Recall(c)} \tag{4.1}$$

$Precision(c)$ is the fraction of correct predictions for c, according to Equation 4.2, where $tpr(c)$ is the true positive rate for c and $fpr(c)$ the false positive rate. $Recall(c)$ is the fraction of instances of c that were correctly predicted (Equation 4.3), where $fnr(c)$ is the false negative rate.

$$Precision(c) = \frac{tpr(c)}{tpr(c) + fpr(c)} \tag{4.2}$$

$$Precision(c) = \frac{tpr(c)}{tpr(c) + fnr(c)} \tag{4.3}$$

On the other hand, micro-averaged F_1 consists of the global harmonic mean between precision and recall, considering all classes $C = \{c_1, c_2, ..., c_k\}$, which is defined as:

$$MicroF_1(c) = \frac{2 \times Precision(C) \times Recall(C)}{Precision(C) + Recall(C)} \tag{4.4}$$

As $MicroF_1$ aggregates the contributions of all classes to compute the average metric, and the value tends to be dominated by the classifier's performance of the most frequent classes. In contrast, the value of $MacroF_1$ is more influenced by the performance on rare classes. Thus, $MacroF_1$ is more suited for imbalanced datasets.

For regression, we compute Pearson and Spearman's rank correlation between the cosine similarity of the sentence embeddings and the ground truth. A high correlation between cosine similarity and the ground truth means that the model is generating good representations for the input text. On the other hand, a low correlation means that the representations produced by the model are badly suited for the dataset. The Pearson correlation between x and y values id defined as:

$$r = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum (x_i - \overline{x})^2 \sum (y_i - \overline{y})^2}} \tag{4.5}$$

where $\overline{x}$ and $\overline{y}$ represent the mean of the values of the x-variable and y-variable, respectively. Spearman's rank correlation is defined as:

$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)} \tag{4.6}$$

where d_i is the difference between the two ranks of each observation and n is the number of observations.

4.3 Parameter Tuning

We have tuned the sentence representations methods based on neural networks considering a set of hyperparameters. As neural networks have many hyperparameters to be tuned, and it is computationally infeasible to use a grid-search or a random-search over all of them. We have chosen a few hyperparameters to be tuned in the considered methods. Table 4.8 shows the range we select for each method and hyperparameter. Each range shown in the table was chosen in a trial-and-error process, where the best set was chosen to perform tuning.

Table 4.8: Hyperparameters of the embedding methods.

Methods	**Hyperparameters**	**Range**
All	Optimizer	Adam, AdamW, Adagrad, SGD
	learning rate	1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 2×10^{-5}
	Dropout prob.	0.0, 0.1, 0.3, 0.5, 0.7
	batch size	16, 32, 64
InferSent (BiLSTM + max-pooling)	Hidden dim.	512, 1024
CNN	Number of filters per layer	128, 256, 512
SBERT	Hidden dropout prob.	0.0, 0.1, 0.3, 0.5, 0.7

During the optimization process, drawn in Figure 4.1, we defined 20 trials of different sets of parameters. Thus, 20 different models were created in order to find the best model. The optimization process chooses the best set of parameters for each dataset and method using cross-validation (CV). At the beginning of this process several parameters are given and they can be randomly chosen to create new models. We use the Hyperopt[2] library for tuning, which implements an algorithm called Tree of Parzen Estimators (TPE). It basically searches for the best set of parameters that minimizes a loss function, which in our case is -1 $\times$ macro-averaged F_1. Figure 4.1 illustrates how the process works. A dataset is split into 5 folds for training and 5 folds for testing in a stratified manner, which guarantees the same distribution of classes in all folds. At the end, we measure the mean of the macro-averaged F_1 (our loss function) of each CV across 20 trials and obtain the highest value to perform the final model.

4.4 Dealing with Numbers

Wallace et al. [2019] showed that BERT has a limited amount of numerical reasoning ability when restricted to numbers of small magnitude. Zhang et al. [2020] proposed to

[2]urlhttp://hyperopt.github.io/hyperopt/

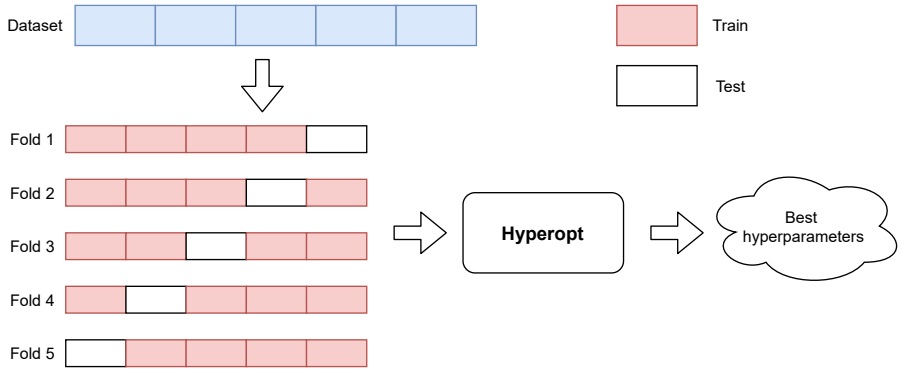


Figure 4.1: Tuning process of the selected methods.

replace every instance of a number in the data with its representation in scientific notation, a combination of an exponent and mantissa: 314.1 is represented as 3141[EXP]2 where [EXP] is a new token introduced into the vocabulary. According to them, this strategy enables BERT to associate objects in the sentence directly with the magnitude expressed in the exponent more easily.

As an attempt to examine and improve the effect of numbers for sentence representations, in particular for item descriptions, we follow a similar approach but evaluating its effectiveness on semantic textual similarity tasks. We also replace the numbers with their scientific notation, but we do not introduce a new token to represent the exponent. For example, 0.000000123 can be written as 1.23e-07, where "e" represents the exponent. Table 4.9 illustrates examples of item descriptions after the application of this approach.

Table 4.9: Example of item descriptions after canonicalizing numbers.

Before	**After**
electrical wire 718 c	electrical wire 7.18e+02 c
triplex cable 0.00455 hm	triplex cable 4.55e-03 hm

Table 4.10 shows results for the synthetic dataset considering both cases, with and without replacing numbers with their scientific notation. We can clearly see that canonicalizing number has a huge effect on the results, not only for SBERT model. Therefore, all the subsequent results presented in this dissertation were obtained considering this operation.

Table 4.10: $MicroF_1$ and $MacroF_1$ results on the English synthetic dataset after canonicalizing numbers.

Model	Synthetic-en (canonicalizing numbers)		Synthetic-en	
	microF1	**macroF1**	**microF1**	**macroF1**
CNN	**78.54**	**73.50**	65.41	55.02
InferSent	**81.05**	**76.79**	65.01	54.63
SBERT	**97.46**	**97.03**	92.54	90.95

4.5 Regression Results

State-of-the-art methods often learn a regression function that maps sentence embeddings to a similarity score. In this study, we use the cosine-similarity to compare the similarity between two sentence embeddings. We also conducted experiments with other distances, such as Manhattan and Euclidean, but the results for all approaches remained roughly the same. The results of Pearson (r) and Spearman correlation (ρ) are depicted in Table 4.11. Performance is reported by convention as $r \times 100$ and $\rho \times 100$. The results in bold represent the best of all methods. For SBERT, we experimented with two setups: training the embeddings on the target dataset, or training the embedding on NLI and then training on the target dataset. As Reimers and Gurevych [2019], we observe that the later strategy leads to a slight improvement of correlations.

The results show that directly using the average of word embeddings leads to rather poor performances when compared to the results obtained by more sophisticated algorithms, such as InferSent and SBERT. The results of average word embeddings are even worse for the synthetic datasets, where the most relevant terms of the sentence are the numbers and the units of measure. In general, the best results were obtained by SBERT, which is the current state-of-the-art method for common sentence similarity tasks. It is also important to highlight that even InferSent and CNN were not able to achieve good results for the synthetic datasets. We argue that word embeddings are not capable of capturing the information about the scalar magnitude of objects and understanding the relationship between two units of measure of the same physical quantity. We can also observe that the bag-of-words model obtained better results than InferSent for STS benchmark and the SICK dataset. However, this strategy achieves poor results for the two synthetic datasets.

4.6 Classification Results

The results of different representations for sentence similarity over a classification perspective can be found in Table 4.12. For performance reasons, we did not run bag-of-

Table 4.11: Pearson correlations (r) and Spearman correlations (ρ) for STS tasks and synthetic datasets.

Model	SICK-R		STSb		Synthetic-en		Synthetic-pt	
	r	ρ	r	ρ	r	ρ	r	ρ
Avg. word2vec embeddings	76.43	65.76	67.57	63.54	-	-	-	-
Avg. GloVe embeddings	75.53	63.75	69.51	68.05	0.64	0.14	-0.59	-0.22
Avg. fastText embeddings	75.91	65.26	70.04	67.03	4.01	4.40	8.20	7.86
Bag-of-words (tf-idf)	81.18	75.53	69.82	69.09	0.31	-0.31	-1.20	-0.43
CNN	78.32	69.26	62.48	62.15	11.01	11.29	15.82	15.40
InferSent	81.54	73.00	63.81	63.13	19.45	19.42	43.77	42.99
SBERT (target dataset)	88.66	83.77	**84.44**	83.70	70.67	70.49	70.41	70.17
SBERT (NLI-target dataset)	**88.90**	**84.59**	84.19	**84.99**	**71.63**	**71.57**	**70.85**	**70.77**

words for the NLI dataset, which has the largest vocabulary, with 84,423 words. We also did not run word embeddings averaging for the synthetic dataset due to the lack of pre-trained models available online and the necessity of fine tuning the embedding on the datasets. As we did not find any word2vec pre-trained models to fine-tune on domain-specific data, we do not report the results of word2vec embeddings averaging on the synthetic datasets. However, we report the results using fastText, which is more robust than word2vec, since it considers ngrams to train the model and build the word vectors.

Table 4.12: F1-score results for the NLI, amazon-walmart and synthetic datasets.

Model	NLI		Amazon-walmart		Synthetic-en		Synthetic-pt	
	microF1	macroF1	microF1	macroF1	microF1	macroF1	microF1	macroF1
Avg. word2vec embeddings	55.29	55.09	71.51	53.80	-	-	-	-
Avg. GloVe embeddings	56.43	56.43	75.73	65.15	69.94	41.15	70.29	41.28
Avg. fastText embeddings	54.57	54.38	68.17	44.92	74.29	57.16	76.96	66.11
Bag-of-words (tf-idf)	-	-	71.51	65.95	69.94	41.15	70.29	41.28
CNN	61.89	61.78	85.47	82.48	78.66	73.59	74.59	68.69
InferSent	67.30	67.26	**87.96**	**85.04**	81.15	76.88	85.31	81.99
SBERT-base	**79.36**	**79.31**	78.63	74.98	97.29	96.83	**97.56**	**97.12**
SBERT-NLI-base	-	-	78.78	74.98	**97.66**	**97.24**	97.49	97.04

As in the case of regression, we observe that simple algorithms, such as bag-of-words and word embeddings averaging, give poor results in comparison to more sophisticated methods. This is particularly true for the synthetic datasets, where the simple strategies lead to low $MacroF1$ score, which suggests that the model cannot predict well in general. Whereas the best results for the NLI and synthetic datasets were obtained by SBERT, the best results for the amazon-walmart dataset were obtained by InterSent (BiLSTM + max polling). This may be due to the higher complexity of SBERT over InferSent, as InferSent is based solely on the BiLSTM architecture, which in turn is easier to tune. Overall, training SBERT first on NLI did not show significant improvements.

Chapter 5

Text Representation Enhancement

Currently, bidirecional encoder representations from transformers (BERT) have attracted a lot of attention in NLP, as they achieve state-of-the-art results on a variety of tasks. Even though BERT is effective on many NLP tasks, it ignores a general linguistic fact, that is, different sentence components serve diverse roles in the meaning of a sentence. For example, in "standard" sentences, it is known that the most crucial parts are the subject, predicate and object, as they represent the primary meaning of a sentence. In addition, words in a sentence are also related to each other by syntactic and semantic relations. In this context, this chapter investigates and proposes a new strategy based on BERT to enhance sentence vectors for supervised and unsupervised tasks.

The proposed strategy divides the original description or sentence into two parts: one refers to the complete text, while the component-focused includes the terms that play a major role in the meaning of the sentences. For item descriptions, these terms can be the number and units of measure, since they carry information about the scalar magnitude of the objects. For other types of sentences, such as sentences and paragraphs, the component-focused part can comprise terms that are related to specific part-of-speech tag or named-entity. In this case, instead of focusing in numbers and units of measure, the model can focus on terms that are person names, quantities, ordinals or cardinals.

5.1 Model Architecture

Our model is deeply inspired by CF-BERT [Yin et al., 2020], a model created to focus on crucial components and syntactic relations of a sentence to get a more powerful representation that yields better performance in downstream natural language processing

(NLP) tasks. While Yin et al. [2020] explores only the main syntactic relations of a sentence, extracting the subject, predicate and object, we also consider other types of words to enhance the sentence representation. As one of the main objectives of this study is to find better representations for product descriptions, we consider the most important components for this context. We explore the fact that items descriptions do not follow the same syntactic structure of a standard sentence or paragraph, and also have information about the scalar magnitudes of objects. Thus, we argue that numeric terms and units of measure serve a crucial role in the meaning of item descriptions, and can have a strong influence on the quality of vector representations in this context.

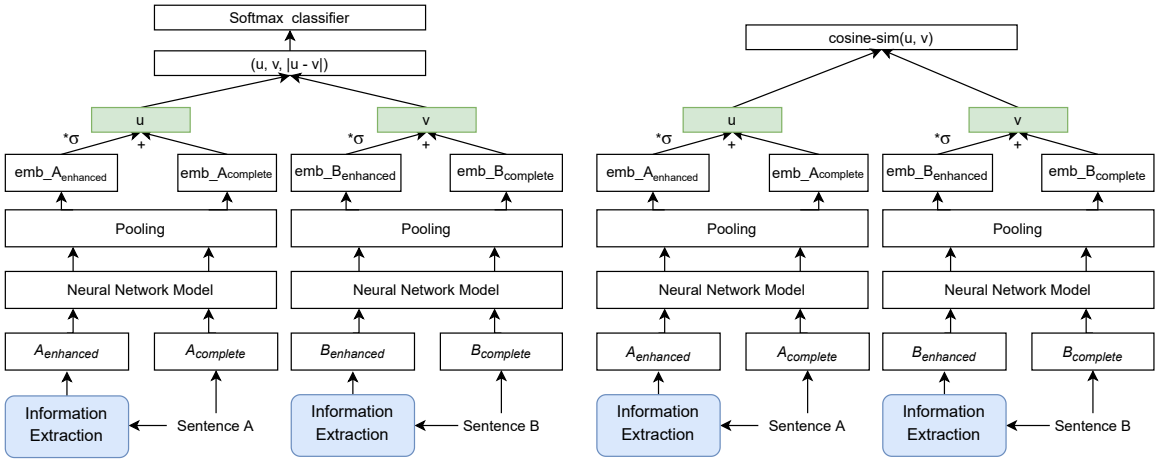


Figure 5.1: Enhancement architecture for classification (left) and regression (right).

The structure of the enhancement representation model is depicted in Figure 5.1. First, the two input sentences are passed to the model to generate fixed-size sentence embeddings considering the whole text. In parallel, the model performs information extraction for each sentence to obtain their component-enhanced parts and generate their embeddings. Hence, for each input sentence we have two vectors: the embeddings for the complete sentence $E_{complete}$ and the embedding for the component-enhanced part $E_{enhanced}$. For each sentence we generate the final embedding E as:

$$E = \sigma \times E_{enhanced} + E_{complete} \tag{5.1}$$

where σ is a weight factor that adjusts the ratio of the component-enhanced part embedding to generate the final sentence representation.

5.2 Information Extraction

As previously mentioned, the component-enhanced part of the model carries the crucial information in the sentence. For extracting the most important components of a sentence we applied two of the most frequently used NLP tasks: part-of-speech tagging (POS tagging) and named-entity recognition. For both tasks we used the library Spacy[1], which provides models pre-trained on the OntoNotes corpus[2].

POS tagging associates words in a text with their corresponding part of speech (tag), based on both its definition and its context. The POS-tagging tool used includes 20 tags: adjective (ADJ), adposition (ADP), adverb (ADV), auxiliary (AUX), conjunction(CONJ), coordinating conjunction (CCONJ), determiner (DET), interjection (INTJ), noun (NOUN), numeral (NUM), particle (PART), pronoun (PRON), proper noun (PROPN), punctuation (PUNT), subordinating conjunction (SCONJ), symbol (SYM), verb (VERB), other (X), end of line (EOL) and space (SPACE). As we focus on the context of product/item descriptions, nouns tend to be more relevant to identify the object in question than other tags.

Named-entity recognition (NER) consists of locating and classifying named entities present in a text into pre-defined categories, such as persons names, organizations, locations, quantities and percentages. Unlike Yin et al. [2020], in the context of our task, we argue that some named entities can correspond to the most crucial part of a sentence, even if they are not present in the subject, predicate or object. The named-entity recognition tool used is able to identify 19 named entities. Among them, we list:

- PERSON: People, including fictional.
- NORP: Nationalities or religious or political group.
- PRODUCT: Objects, vehicles, foods, etc.
- LANGUAGE: Any named language.
- PERCENT: Percentage, including "%".
- QUANTITY: Measurements, as of weight or distance.
- ORDINAL: first, second, etc.
- CARDINAL: Numerals that do not fall under another type.

[1] `https://spacy.io`
[2] `https://catalog.ldc.upenn.edu/LDC2013T19`

It is very important to highlight that different than Yin et al. [2020], we do not consider the subject, object and predicate of the sentence for the component-enhanced part of the representation. For STSb and SICK-R, in particular, the component-enhanced part, extracted by the information extraction step, consists of nouns and terms that are part of one of the entities: PERSON, QUANTITY, ORDINAL, or CARDINAL. On the other hand, for the synthetic datasets, the component-enhanced part of the representation corresponds to numbers and units of measure, since the instance consists of item descriptions. The same applies to the Amazon-Walmart dataset, in which we extracted numbers and units of measure as well as the terms that are part of the entities: PRODUCT, QUANTITY, ORDINAL and CARDINAL. Some examples of the two parts of the representation - base (S_{base}) and component-enhanced (S_{cf}) for each dataset are listed in Table 5.1.

Table 5.1: Examples of component-enhanced part for sentences and item descriptions.

Dataset	S_{base}	S_{cf}
Synthetic-en	gasoline 792 in3	792 in3
	lighting system 249 watt	249 watt
	external hd 70.8 tb	70.8 tb
	fluorescent lamp 502 kw	502 kw
	blue paper roll 644 in	644 in
	cement 1.48 lb	1.48 lb
SICK-R	A group of kids is playing in a yard and an old man is standing in the background	group kids yard man background
	The young boys are playing outdoors and the man is smiling nearby	boys man
	Two dogs are fighting	Two dogs
	A woman is wearing an Egyptian hat on her head	woman hat head
	A hiker is on top of the mountain and is dancing	hiker top mountain
	The black woman is wearing glasses over the headdress	woman glasses headdress
Amazon-walmart	lorex vq2121 100 ft high intensity night vision ir illuminator	100 ft
	slappa sl nsv 122 10 inch lady damask netbook sleeve black black	122 10 inch
	hp envy 100 e all in one d 410 a printer cn 517 a b 1 h	hp 100 in d 410 517 1 h
	belkin 6 firewire cable 4 pin 6 pin	6 4 6
	trendnet 2 port usb kvm switch kit with audio includes 2 x kvm cables tk 209 k blue	2 2 209 k

5.3 Experimental Evaluation

To evaluate the results of the proposed strategy, we considered the datasets described in Section 4.1, except NLI, since it has already been used to train the models. In Chapter 4, we observed that by first training SBERT on NLI and then on specific NLP tasks yields better results than only training SBERT on the specific NLP tasks. Therefore, in this section we only report the results using this approach. In other

words, the models used were instantiated from pre-trained BERT model fine-tuned with mean-tokens pooling on the NLI dataset. The results were obtained by training the models with 5 random seeds. Experiments were repeated 5 times with a 5-fold cross validation, totalizing 25 results per experiment. To compare the average results of our repeated experiments, we assess their statistical significance by applying a paired t-test with 95% confidence.

The proposed strategy requires an additional parameter, the weight factor σ. Its value was set to 1.0, 0.5 and 0.2 for the Portuguese synthetic dataset, English synthetic dataset and STS datasets (STSb and SICK-R), respectively, after performing a grid search.

5.3.1 Regression Task

As in Chapter 4, we use both the Pearson correlation and Spearman correlation as similarity correlation measures to evaluate the results, reported in Table 5.2. Results are reported by convention as $r \times 100$ and $\rho \times 100$, and numbers in bold represent the best of all methods. Table 5.2 indicates that the addition of the component-enhancement representation has improved the quality of the representation in the synthetic datasets. For the English and Portuguese synthetic datasets, the Pearson correlation improved by approximately 1.27 and 1.51, respectively, compared to the results obtained by SBERT. This can be explained by the fact that, for the synthetic datasets, numbers and units of measure play the most important role in the meaning of item descriptions. The same was not observed for the STS tasks, where there was no improvement in comparison to the results obtained by SBERT. However, we expect the representations obtained by SBERT fine-tuned with information extraction for item descriptions to be more robust than those obtained by SBERT fine-tuned only on the NLI dataset. It is important to observe that the results achieved by SBERT with enhancement are just slightly worse than those achieved by SBERT, which shows that the enhancement can be used without significant loss of performance.

Differently, Yin et al. [2020] observed that CF-BERT (first training of NLI, then training on specific dataset via component focusing) was able to achieve better results on STSb and SICK-R datasets, when using dependency parsing to get the component-enhanced part. This highlight the importance of the subject, object and predicate for the meaning of a sentence, which is not necessarily true for item descriptions.

Table 5.2: Pearson correlations (r) and Spearman correlations (ρ) results on the STS tasks and on the synthetic datasets when using enhancement. Results in bold are the best according to the statistical tests.

Dataset	Metric	Model	
		SBERT	**Enhancement**
SICK-R	r	**88.90 ± 0.04**	88.71 ± 0.03
	ρ	**84.51 ± 0.06**	84.20 ± 0.03
STSb	r	84.22 ± 0.16	84.00 ± 0.06
	ρ	85.03 ± 0.16	84.81 ± 0.09
Synthetic-en	r	71.71 ± 0.11	**72.98 ± 0.22**
	ρ	71.69 ± 0.13	**72.95 ± 0.24**
Synthetic-pt	r	71.11 ± 0.18	**72.62 ± 0.18**
	ρ	71.02 ± 0.18	**72.61 ± 0.19**

5.3.2 Classification Task

Table 5.3 shows the results of $MicroF_1$ and $MacroF_1$ on the Amazon-walmart dataset and on the synthetic datasets. The application of the enhancement on the Amazon-walmart dataset achieved an improvement of 1.75 for $MicroF_1$ and 2.08 points for $MacroF_1$. Besides the Amazon-walmart dataset, SBERT model with enhancement did not achieved a significant improvement over SBERT. However, it is important to highlight that SBERT has already achieved good results on the synthetic datasets, with both $MicroF_1$ and $MacroF_1$, having values close to 100. In this case, improving further the results is much more difficult than in the Amazon-walmart dataset. In addition, we can argue that the enhancement on Amazon-walmart achieved better results due to the characteristic of the dataset, which is very small and has a lot of information about the considered categories for the component-focused part of the model.

Table 5.3: $MicroF_1$ and $MacroF_1$ results on the Amazon-walmart dataset and on the synthetic datasets when using enhancement. Results in bold are the best according to a paired t-test.

Dataset	Metric	Model	
		SBERT	**Enhancement**
Amazon-walmart	microF1	80.09 ± 1.77	**81.84 ± 1.94**
	macroF1	76.13 ± 2.51	78.21 + 2.12
Synthetic-en	microF1	97.29 ± 0.18	97.56 ± 0.05
	macroF1	96.83 ± 0.20	97.14 + 0.06
Synthetic-pt	microF1	**97.50 ± 0.10**	97.49 ± 0.06
	macroF1	**97.05 ± 0.12**	97.03 ± 0.07

Chapter 6

Evaluation of Unique Objects Clustering

In this chapter we describe the results of our experiments on text clustering with different sentence representations: bag-of-words, word embeddings averaging, SIF (smooth inverse frequency), Sent2vec, CNN, InferSent, Sentence-BERT and the method proposed in this study (presented in Chapter 5). We considered the real-word dataset extracted from the public ministry of the state of Minas Gerais in Brazil.

6.1 Experimental Setup

Table 6.1 presents a summary of a subset of experiments performed by instantiating the proposed methodology. Cells that have a dash ("-") represent situations where hyperparameters do not apply to the strategy. In order to find the best sentence representation method and clustering algorithm for item descriptions, we conducted a varied range of experiments. For each sentence representation model, we selected the most appropriate pooling strategy and embedding size. Most experiments considered the HDBSCAN algorithm, as it gave the best results for our dataset. We also show the results of X-Means after grouping the description by their first token, in order to compare the results with those obtained by HDBSCAN.

6.1.1 Evaluation Metrics

Four evaluation metrics were considered to assess the quality of the groups obtained by each clustering strategy depicted in Table 6.1. The first metric measures the percentage of items (i.e., descriptions) considered outliers. A low percentage of noise (i.e., outliers)

Table 6.1: Clustering experiments.

Algorithm	Representation	Pooling Strategy	Embedding Size
HDBSCAN	SBERT-NLI-synthetic_pt	Mean	728
HDBSCAN	EnhancementBERT-NLI-synthetic_pt	Mean	728
FToken + HDBSCAN	GloVe emb. averaging	Mean	300
FToken + HDBSCAN	fastText emb. averaging	Mean	300
FToken + HDBSCAN	bag-of-words	-	Vocabulary size
FToken + HDBSCAN	SIF	Weighted average	300
FToken + HDBSCAN	Sent2vec	-	700
FToken + HDBSCAN	CNN	Mean	364
FToken + HDBSCAN	InferSent	Max	1,024
FToken + HDBSCAN	SBERT-NLI	Mean	728
FToken + HDBSCAN	SBERT-NLI-synthetic_pt	Mean	728
FToken + HDBSCAN	EnhancementBERT-NLI-synthetic_pt	Mean	728
FToken + X-means	SBERT-NLI-synthetic_pt	Mean	728

indicates good grouping quality. For HDBSCAN, in particular, instances considered noise were assigned to group "-1". Noises are instances that cannot be attributed to any of the groups found by HDBSCAN, as they are very different in comparison to the other instances.

The second metric is the Davies-Bouldin score, defined as the average similarity of each group with its most similar group. The similarity of a group is a ratio between intra-group distances to inter-group distances, defined by the pairwise distance between centers belonging to different groups. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering.

The Calinski and Harabasz score was also considered, as it assesses the ratio between intra-group dispersion and inter-group dispersion. It is also known as the Variance Ratio Criterion. The higher its value, the better the groups generated by the clustering algorithm. Basically, high values of this measure indicate that the groups are dense and well separated.

Finally, we report the Silhouette Coefficient, which measures the cohesion and separation of groups. It is based on the difference between the distance an instance has from other instances in its cluster and from other clusters. This coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is $s_i = (b - a)/\max(a, b)$, where:

- a is the average distance between a point and all other points in the same group;

- b is the distance between a point and the nearest group to which the point does not belong.

The silhouette Coefficient is defined as the average of the values s_i over all items. Note that this coefficient is only defined if the number of clusters is greater than or equal to two and less than or equal to the number of samples minus one. The Silhouette Coefficient value of an item is in the range [-1, 1]. The best value is 1 and the worst value is -1. A silhouette coefficient close to 1 indicates that the instance x_i is much closer to items in its own cluster and is far from other clusters. A silhouette coefficient close to zero indicates that x_i is close to the boundary between two clusters, indicating overlapping clusters. Finally, a negative value indicates that a sample has been assigned to the wrong cluster, since a different cluster is more similar.

6.1.2 Setting Hyperparameters

Apart from the parameter *min_cluster_size*, which controls the minimum size of clusters, the parameters *min_samples* and *metric* of HDBSCAN were analyzed. *min_samples* defines the number of samples in a neighbourhood for a point to be considered a core point, and *metric* the metric to be used to calculate the distance between the points.

The minimum size of clusters was set to 30 after analysing Figure 6.1, where we observed that 3.90% of the groups generated by the first token grouping have less than 31 items.

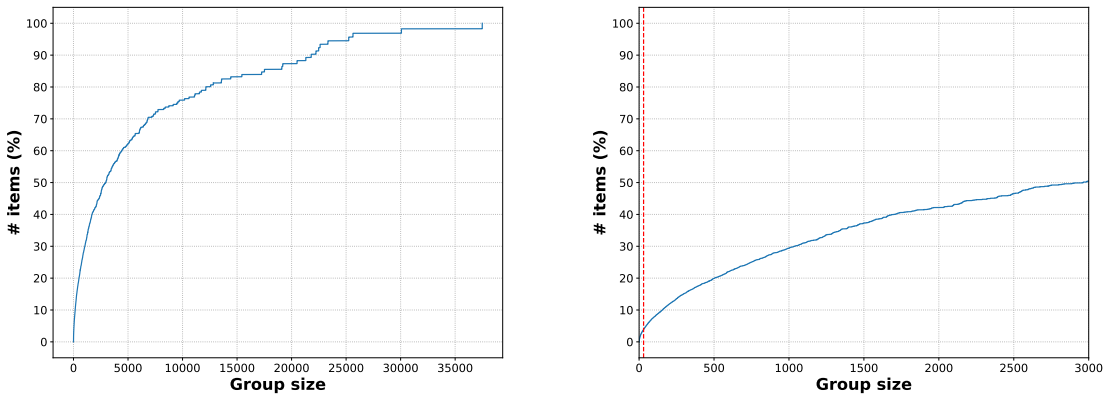


Figure 6.1: Cumulative Distribution of the number of items by group size for the groups obtained by the first tokens. The figure on the left represents the complete distribution. The figure on the right represents a zoom up to 3000. The red vertical line represents group size 30.

For the minimum number of samples (*min_samples*), the higher its value the more conservative is the clustering and the greater the amount of noise generated. In this way, as the value of *min_samples* increases, the groups become denser. It was possible to observe that indeed the smaller the value of *min_samples*, the smaller the number of items classified as outliers (noise) and the worse the quality of the groups found, since they have more points distant from the core points. Thus, we chose to keep the default value for the parameter *min_samples*, which is None. The metric parameter was kept as Euclidean, since the vectors are normalized.

6.2 Experimental Results

From the experiments depicted in Table 6.1, it is possible to identify the best clustering strategy and the best sentence representation method for our dataset. The results for each clustering strategy and sentence representation model are reported in Table 6.3, and the evaluation of the clusters for each approach is depicted in Table 6.2.

It is possible to observe that the best results for the silhouette coefficient calculated using cosine distance were obtained when the item representations were built from the SIF model and when the first token grouping was applied (Silhouette coefficient of 0.836). On the other hand, the best results for the silhouette coefficient using Euclidean distance were obtained by the description generated by the SBERT model with enhancement, first trained on the NLI dataset and fine-tuned on the Portuguese synthetic dataset of item descriptions. When only applied to HDBSCAN, this representation obtained the best results for the Calinski and Harabasz score and the Davies-bouldin score (Calinski and Harabasz score of 1,452,345.10 and Davies-boulding score of 0.37). We believe this is due to the high number of items categorized as outliers (43.14%), which are removed from the original collection of items for evaluating the clustering results. As we observed for the semantic textual similarity tasks, discussed in Chapter 4, the SBERT model first trained on the NLI datasets leads to better results in general.

For the enhancement SBERT model, we considered units of measure and numbers, as they play a crucial role in the meaning of the item descriptions, carrying information about the scalar magnitude of objects. It is possible to note that the SBERT model with enhancement presents no improvement in comparison to the results obtained by SBERT, when the first token clustering is applied. In contrast, the enhancement model achieved better results in the case where only HDBSCAN is executed. It is also important to highlight that fine-tuning SBERT on the Portuguese-synthetic dataset improves

the results for all metrics. This shows the importance of fine-tuning the sentence representation model on domain-specific data before deriving vectors for unsupervised tasks like text clustering.

Even though simple text representation strategies performed worse than SBERT, they lead to reasonable results. For the silhouette coefficient calculated using cosine distance, the best results were obtained by the SIF model (Silhouette coefficient of 0.836), which represents a sentence by a weighted average of word vectors in an unsupervised manner. As shown by [Arora et al., 2017], the SIF model can generate better sentence representations than sophisticated supervised methods including CNN and InferSent. These methods gave the worst results considering all metrics. In addition, although the bag-of-words representation is very simple, it still presents better results than those obtained by CNN and InferSent. It is important to remember that SIF gives more importance to words that have a low frequency in the dataset. As the collection of item descriptions comprises several frequent numbers, they can have little importance in the representation. This highlights that there is a trade-off between performance and given importance to some tokens, such as number and units of measure, which confirms that it is very difficult to generate robust vectors for terms that carry numeracy information.

Table 6.2: Evaluation of clusters found by each clustering strategy.

Algorithm	**Representation**	**Metrics**			
		Avg. Calinski	**Avg. Davies-bouldin**	**Avg. Silhouette-cosine**	**Avg. Silhouette-euclidean**
HDBSCAN	SBERT-NLI-synthetic_pt	1,236,776.30	0.39	0.72	0.61
HDBSCAN	EnhancementBERT-NLI-synthetic_pt	**1,452,345.10**	**0.37**	0.754	0.641
FToken + HDBSCAN	GloVe emb. averaging	1,986.6	0.475	0.79	0.628
FToken + HDBSCAN	fastText emb. averaging	2,295.4	0.45	0.802	0.643
FToken + HDBSCAN	bag-of-words	2,018.6	0.48	0.787	0.621
FToken + HDBSCAN	SIF	2,730.4	0.415	**0.836**	0.679
FToken + HDBSCAN	Sent2vec	2,597.5	0.462	0.798	0.635
FToken + HDBSCAN	CNN	1,836.2	0.476	0.785	0.621
FToken + HDBSCAN	InferSent	1,742.8	0.48	0.783	0.618
FToken + HDBSCAN	SBERT-NLI	2,360.5	0.445	0.799	0.641
FToken + HDBSCAN	SBERT-NLI-synthetic_pt	3,565.9	0.421	0.828	**0.685**
FToken + HDBSCAN	EnhancementBERT-NLI-synthetic_pt	3,382.2	0.434	0.817	0.672
FToken + X-means	SBERT-NLI-synthetic_pt	2,440.2	0.643	0.73	0.566

Regarding the clustering results, it is possible to observe that HDBSCAN leads to a smaller number of groups than X-means, while also identifying outliers. We can see that when only HDBSCAN is executed the number of clusters is much smaller in comparison to the other strategies. However, the number of items identified as outliers is much larger. This strategy leads to 5,160 groups and 45.02% of outliers when SBERT model is used to derive vectors for the item descriptions. We can also see that the results for the silhouette coefficients are significantly worse in comparison

to when the first token grouping is used. We believe that silhouette coefficients are the best metrics to measure the quality of clustering results as they consider the mean intra-cluster distance and the mean nearest-cluster distance of all samples. Therefore, we conclude that the application of a simple grouping strategy before the execution of a clustering algorithm leads to better results. It is possible to better separate the items in the dataset, with a significant improvement in the quality of the clusters found by the first token grouping.

Table 6.3: Clustering results.

Algorithm	**Representation**	**# groups**	**outliers (%)**	**excluded (%)**
HDBSCAN	SBERT-NLI-synthetic_pt	5,160	45.02	45.02
HDBSCAN	EnhancementBERT-NLI-synthetic_pt	5,925	43.14	43.14
FToken + HDBSCAN	GloVe emb. averaging	33,565	22.62	26.52
FToken + HDBSCAN	fastText emb. averaging	33,278	20.13	24.03
FToken + HDBSCAN	bag-of-words	31,255	25.89	29.79
FToken + HDBSCAN	SIF	36,086	23.58	27.48
FToken + HDBSCAN	Sent2vec	32,415	19.78	23.68
FToken + HDBSCAN	CNN	32,620	23.83	27.73
FToken + HDBSCAN	InferSent	32,572	**27.59**	**31.49**
FToken + HDBSCAN	SBERT-NLI	33,115	20.31	24.21
FToken + HDBSCAN	SBERT-NLI-synthetic_pt	36,644	17.84	21.74
FToken + HDBSCAN	EnhancementBERT-NLI-synthetic_pt	**36,800**	20.00	23.9
FToken + X-means	SBERT-NLI-synthetic_pt	91,569	0.00	3.90

In summary, we can see through this analysis of text clustering that there is no significance using sophisticated supervised methods, such as CNN and InferSent, besides SBERT, to derive vectors for unsupervised tasks. On the other hand, unsupervised methods like SIF can give results comparable to those obtained by SBERT, which highlights the efficiency of a simple strategy of weighted average for building sentence representations, despite giving less importance for frequent tokens. We also confirm that naive strategies, such as bag-of-words and word embeddings averaging lead to worse results than sophisticated methods, such as SIF, Sent2vec and SBERT. Moreover, we clearly see that fine-tuning the SBERT model on the synthetic dataset gives more suitable vector to item descriptions in comparison to other strategies, since it achieves better results according to the clustering evaluation metrics on the real-world dataset.

6.2.1 Qualitative Analysis

This section presents a qualitative analysis of the groups of items obtained by applying HDBSCAN after the first token grouping, using the representation obtained by SBERT trained on the NLI dataset, then fine-tuned on the Portuguese synthetic dataset. This analysis embraces both general and specific characteristics of groups.

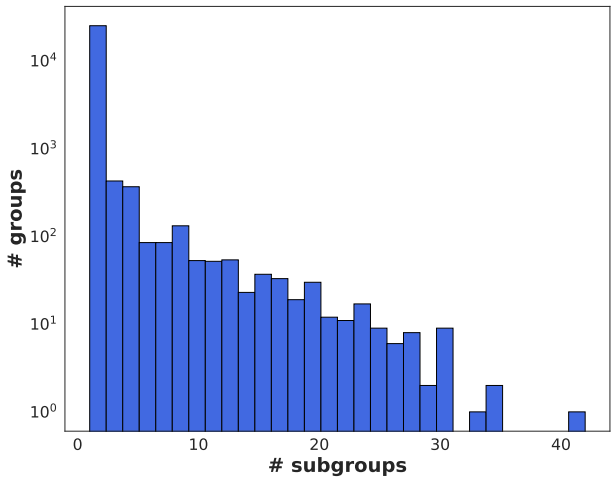


Figure 6.2: Histogram of the number of subgroups generated by first token grouping.

Figure 6.2 shows the distribution of the number of subgroups generated from the execution of HDBSCAN for each group obtained by the initial grouping. We can observe that most groups were divided into a few subgroups. This is expected, since HDBSCAN was not applied in most groups generated by the initial grouping. However, we can note that a significant portion of the groups was split into 20 subgroups or more. This is the case of the "caneta", "broca", "lampada", "disjuntor" and "tubo", which comprises a large collection of items. After the execution of HDBSCAN, these groups were divided into 31, 31, 34, 31 and 33 subgroups, respectively. This demonstrates that the application of a clustering algorithm after the initial grouping manages to separate items from generic groups into smaller and more specific subgroups.

We also analyze some characteristics of specific groups, selected arbitrarily based on a collection of products that are frequently searched by content analysts. The groups are related to the following topics: i) fuels and lubricants, ii) hospital and medicines and iii) automotive parts. Table 6.4 presents examples of items for each topic. For each group, we can see the number of items and the number of subgroups obtained by the application of HDBSCAN.

Table 6.4: Examples of groups: (i) fuels, (ii) hospital items and medicines, and (iii) automotive parts.

First token	**# items**	**# subgroups**
alcool	5,214	16
querosene	84	2
oleo	19,339	18
luva	19,370	23
mascara	2,331	16
acido	3,738	22
sabonete	3,253	11
vitamina	699	8
dipirona	1,441	10
atenolol	370	5
estetoscopio	167	3
reanimador	228	5
pneu	16,700	25
freio	668	2
parabrisa	457	2
parachoque	123	2
roda	297	5
lampada	15,473	33
bateria	3,184	21
amortecedor	1,886	12

Table 6.5: Subgroups of "Máscara".

Máscara			
Subgroup	**# items**	**Most frequent tokens**	**Example**
mascara_1	490	"mascara", "5.00e+01", "c", "elastico", "descartavel", "triplo", "com", "cirurgico", "cx", "unidade"	mascara cirurgico descartavel caixa c 50 unid
mascara_6	279	"mascara", "1.00e+02", "descartavel", "c", "elastico", "com", "polipropileno", "cirurgico", "unidade", "camada"	mascara cirurgico descartavel c elastico caixa 100 unidade atoxicar
mascara_0	270	"mascara", "com", "para", "elastico", "transparente", "descartavel", "facial", "oxigenio", "alto", "reservatorio"	mascara laringeo usado paciente para controle via aereo respiratorio
mascara_11	234	"mascara", "9.50e+01", "n", "com", "descartavel", "filtro", "elastico", "protecao", "%", "c"	mascara cirurgico protecao n 95 tamanho medio contra bacilo tuberculose
mascara_9	198	"mascara", "3.00e+00", "descartavel", "com", "c", "n", "camada", "laringeo", "para", "3.00e+01"	mascara caruru descartavel cor branco confeccionada falso tecido com 3 camada

Table 6.5 presents the largest subgroups for the first token "Máscara". The group "Máscara" has 2,331 items. The application of HDBSCAN split this group into 16 subgroups, each one representing a specific variation of the product "Máscara". We can see that four subgroups depicted in Table 6.5 have numbers as one of the most frequent tokens, which highlights the importance of numbers for describing items. The

Table 6.6: Subgroups of "Lâmpada".

Lâmpada			
Subgroup	**# items**	**Most frequent tokens**	**Example**
lampada_7	1,475	"lampada", "w", "1.27e+02", "v", "x", "fluorescente", "compactar", "4.00e+01", "1.00e+02", "3.40e+01"	lampada halogenar 100 w 127 v 2020
lampada_22	1,376	"lampada", "1.20e+01", "v", "w", "polo", "1.00e+00", "led", "1.10e+02", "2.00e+00", "2.01e+03"	lampada 1140 12 v fiat unir 2001
lampada_24	1,311	"lampada", "w", "2.50e+02", "v", "2.20e+02", "2.50e+01", "misto", "vapor", "2.40e+01", "x"	lampada h15 240 v placa alf 2570
lampada_9	1,171	"lampada", "w", "1.50e+01", "v", "1.27e+02", "1.50e+02", "fluorescente", "compactar", "x", "led"	lampada piscar doble 1.40 ano 2001
lampada_12	1,079	"lampada", "w", "2.00e+01", "2.20e+02", "v", "fluorescente", "led", "x", "5.00e+02", "2.40e+01"	lampada compactar economico 20 w 2020

same can be observed for the subgroups for the first token "Lâmpada" in Table 6.6.

Chapter 7

Conclusions and Future Work

This dissertation approached a real-word problem related to the automatic detection of overpricing in public procurement, i.e., checking whether the item that is being bought by a public organization is overpriced. One of the main challenges of this problem consists in the fact that only the item descriptions and the prices are available for analysis. In this context, our work proposed a new methodology to solve this problem. Particularly, we modeled this problem as a text clustering problem, where similar items are grouped together based on their descriptions. From the groups of items found, we can later calculate commonly used statistic measures for the item prices, which can indicate whether an item is overpriced or not.

Another problem that comes with text clustering is to find the best sentence representation method to map each instance to a vector in a semantic space. In an attempt to come with a solution, our study comprises an analysis of the most popular sentence representation models, which explore different strategies based on supervised and unsupervised tasks. Some of these methods are the state-of-the-art in well-known NLP tasks like semantic textual similarity (STS). We showed that simple strategies, such as bag-of-words and word embeddings averaging, can lead to unsatisfactory results. Therefore, we argue that the currently used sentence representation algorithms may not be ideal for item descriptions as they are for sentences and paragraphs, since these descriptions do not follow a grammatical structure (subject, object and predicate), and often have numeric terms and units of measure. These types of terms usually carry information about scalar magnitudes of objects. Hence, due to their singular properties, item descriptions offer new challenges in the context of text representation.

Since our real-world dataset comprises item descriptions, our study focused on this context. Trying to improve the quality of representations for item descriptions we proposed a new method based on the CF-BERT model proposed by Yin et al. [2020]. In

summary, the method consists of two major steps: information extraction and training on semantic textual similarity tasks. For the information extraction part, we implemented two well-known NLP tasks named part-of-speech tagging (POS-tagging) and named-entity recognition (NER). The main idea of this step is to focus on the most important tokens for the task being solved. Knowing the part-of-speech tags and named entities, we can focus on this information to enhance the sentence representations using the SBERT [Reimers and Gurevych, 2019] model, a modification of the pre-trained BERT network that use siamese network structure to train and derive semantically meaningful sentence embeddings. As we identified numeric terms and units of measure as being the most important components of item descriptions, we focus on them to enhance the item representations.

For the investigation of the best sentence representation method for item descriptions, we designed synthetic datasets of product titles pairs, which can be used to train the selected models taking into consideration classification or regression objective functions. Together with these datasets, we also considered the STS benchmark, the SICK-Relatedness dataset, the NLI dataset and the Amazon-Walmart dataset to evaluate the sentence representation models. In general, experiments showed that the best method for all datasets consists of SBERT. Experiments also showed that a simple strategy of enhancement can lead to better results in domain-specific datasets like the synthetic dataset of item description pairs that we designed.

Following the exploratory study of sentence representation methods, our work presented a new clustering framework for solving the real-word problem introduced, which can also be used in other tasks. The framework corresponds to the four steps: (i) Text cleaning, (ii) Information extraction, (iii) Text representation, and (iv) Grouping. Regarding the text representation step, the framework covered bag-of-words, word embeddings averaging and pre-trained models trained on common STS tasks, which were used to derive vectors for the item descriptions. For the grouping stage, we focused on X-means and HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise). These algorithms were selected due to their opposite strategies and the fact that we do not need to specify beforehand the number of clusters. While X-means is a centroid-based and flat clustering algorithm, HDBSCAN is hierarchical and density-based. HDBSCAN is, particularly, interesting for our application due to its hierarchical nature and the ability of identifying outliers.

From the clustering results, we can conclude that simple text representation strategies, such as bag-of-words and word embeddings averaging give poor results on our dataset, considering all evaluation metrics. Besides, we also observed that first applying first token grouping, then HDBSCAN, on item descriptions can lead to bet-

ter results in comparison to when only HDBSCAN is used. In addition, we noticed that SIF model obtained results comparable to the state-of-the-art method, and that SBERT gives the best values overall.

7.1 Future Works

As a future work, we suggest a more comprehensive study of the clusters obtained by the proposed framework, the execution and evaluation of other clustering algorithms that do not necessarily follow the same idea of X-means and HDBSCAN and the application of the proposed framework to other real-word datasets in different contexts. Moreover, it might be interesting to explore the pre-trained sentence representation models in other supervised and unsupervised natural language processing tasks.

In order to analyze and fully understand the clusters obtained by the clustering strategy discussed in this dissertation, topic modeling could be a good direction as a next step in the research. We could interpret the clusters as a topic and extract keywords using the semantic space built from pre-trained sentence representation models. By assuming that many semantically similar sentences are indicative of an underlying topic, we can better understand the clustering results and use the framework to semantic search.

Another important direction for our work is exploring the text enhancement methodology for improving the quality of sentence representation in other contexts. In this study, we focused on the context of item description, in which we observed that numbers and units of measure are the most important components, since they carry information about the scalar magnitude of objects. In other contexts, such as product reviews and electronic health records (EHR), the most important components can be different than those considered in this work, and may also be used to enhance the quality of the sentence representations.

Bibliography

Agibetov, A., Blagec, K., Xu, H., and Samwald, M. (2018). Fast and scalable neural embedding models for biomedical sentence classification. *BMC bioinformatics*, 19(1):1--9.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817--1853.

Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2016). paper recommender systems: A literature survey. *International Journal on Digital Libraries*, 17(4):305--338.

Bekkerman, R., El-Yaniv, R., Tishby, N., and Winter, Y. (2001). On feature distributional clustering for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 146–153, New York, NY, USA. Association for Computing Machinery.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120--128.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146. ISSN 2307-387X.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632--642, Lisbon, Portugal. Association for Computational Linguistics.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467--479.

Cadez, I., Heckerman, D., Meek, C., Smyth, P., and White, S. (2003). Model-based clustering and visualization of navigation patterns on a web site. *Data mining and knowledge discovery*, 7(4):399--424.

Campello, R. J. G. B., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G., editors, *Advances in Knowledge Discovery and Data Mining*, pages 160--172, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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*, 12:2493--2537.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In Palmer, M., Hwa, R., and Riedel, S., editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670--680. Association for Computational Linguistics.

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079--3087.

Das, S., G.C., P. S., Doan, A., Naughton, J. F., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V., and Park, Y. (2017). Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 1431–1446, New York, NY, USA. Association for Computing Machinery.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171--4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dong, X., Shen, J., Wang, W., Liu, Y., Shao, L., and Porikli, F. (2018). Hyperparameter optimization for tracking with continuous deep q-learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 518--527.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996a). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996b). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226--231.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1--309.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hartmann, N. S., Fonseca, E. R., Shulby, C. D., Treviso, M. V., Rodrigues, J. S., and Aluísio, S. M. (2017). Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *Anais do XI Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*, pages 122--131, Porto Alegre, RS, Brasil. SBC.

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367--1377, San Diego, California. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328--339, Melbourne, Australia. Association for Computational Linguistics.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2042–2050, Cambridge, MA, USA. MIT Press.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427--431, Valencia, Spain. Association for Computational Linguistics.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655--665, Baltimore, Maryland. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746--1751, Doha, Qatar. Association for Computational Linguistics.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294--3302.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188--1196.

Lee, H., Pham, P., Largman, Y., and Ng, A. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*, 22:1096--1104.

Li, Y., Wang, C., Han, F., Han, J., Roth, D., and Yan, X. (2013). Mining evidences for named entity disambiguation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1070--1078.

Logeswaran, L. and Lee, H. (2018). An efficient framework for learning sentence representations. *International Conference on Learning Representations*.

Luong, M., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R., et al. (2014). A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216--223. Reykjavik.

Masci, J., Giusti, A., Ciresan, D., Fricout, G., and Schmidhuber, J. (2013). A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *2013 IEEE International Conference on Image Processing*, pages 2713--2717. IEEE.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Melamud, O., Goldberger, J., and Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51--61.

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.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of*

Data, SIGMOD '18, page 19–34, New York, NY, USA. Association for Computing Machinery.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 380--390.

Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 528--540.

Pelleg, D. and Moore, A. W. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 727–734, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Pennington, J., Socher, R., and Manning, C. (2014a). 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.

Pennington, J., Socher, R., and Manning, C. D. (2014b). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532--1543.

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

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning. *Technical report, OpenAI*.

Radu, R.-G., Rădulescu, I.-M., Truică, C.-O., Apostol, E.-S., and Mocanu, M. (2020). Clustering documents using the document to vector model for dimensionality reduction. In *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6.

Rahman, W., Hasan, M. K., Lee, S., Bagher Zadeh, A., Mao, C., Morency, L.-P., and Hoque, E. (2020). Integrating multimodal information in large pretrained transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2359--2369, Online. Association for Computational Linguistics.

Rao, A. and Spasojevic, N. (2016). Actionable and political text classification using word embeddings and LSTM. *CoRR*, abs/1607.02501.

Rao, K., Sak, H., and Prabhavalkar, R. (2017). Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982--3992, Hong Kong, China. Association for Computational Linguistics.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533--536.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620. ISSN 0001-0782.

Salvaris, M., Dean, D., and Tok, W. H. (2018). Deep learning with azure. *Berkeley, CA: Apress. doi*, 10:978--1.

Sarker, I. H., Abushark, Y. B., Alsolami, F., and Khan, A. I. (2020a). Intrudtree: A machine learning based cyber security intrusion detection model. *Symmetry*, 12(5). ISSN 2073-8994.

Sarker, I. H., Kayes, A., Badsha, S., Alqahtani, H., Watters, P., and Ng, A. (2020b). Cybersecurity data science: an overview from machine learning perspective. *Journal of Big data*, 7(1):1--29.

Schifano, S. F., Sgarbanti, T., Tomassetti, L., et al. (2018). Authorship recognition and disambiguation of scientific papers using a neural networks approach. *Proceedings of Science*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual*

meeting of the association for computational linguistics, pages 384--394. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998--6008.

Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. (2019). Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307--5315, Hong Kong, China. Association for Computational Linguistics.

Wang, P., Xu, B., Xu, J., Tian, G., Liu, C.-L., and Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174:806–814. ISSN 0925-2312.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2016). Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112--1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69--90.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480--1489, San Diego, California. Association for Computational Linguistics.

Yin, X., Zhang, W., Zhu, W., Liu, S., and Yao, T. (2020). Improving sentence representations via component focusing. *Applied Sciences*, 10(3). ISSN 2076-3417.

Yogatama, D., Kong, L., and Smith, N. A. (2015). Bayesian optimization of text representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2100--2105, Lisbon, Portugal. Association for Computational Linguistics.

Zeng, H.-J., He, Q.-C., Chen, Z., Ma, W.-Y., and Ma, J. (2004). Learning to cluster web search results. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 210–217, New York, NY, USA. Association for Computing Machinery.

Zhang, X., Ramachandran, D., Tenney, I., Elazar, Y., and Roth, D. (2020). Do language embeddings capture scales? *arXiv preprint arXiv:2010.05345*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649--657.