Pontifícia Universidade Católica do Rio de Janeiro

Luis Felipe Müller de Oliveira Henriques

Deep Architecture for Quotation Extraction

Dissertação de Mestrado

Dissertation presented to the Programa de Pósgraduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Ruy Luiz Milidiú

Rio de Janeiro
March 2017

Luis Felipe Müller de Oliveira Henriques

Deep Architecture for Quotation Extraction

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.
Approved by the undersigned Examination Committee.

Rio de Janeiro, March 8th, 2017

Luis Felipe Müller de Oliveira Henriques

Graduated in Computer Science from the Pontifical Catholic University of Rio de Janeiro (PUC-RIO). Joined the LEARN lab in 2015, focusing his research on Machine Learning and Natural Language Processing.

Bibliographic data

Henriques, Luis Felipe Müller de Oliveira

Deep architecture for quotation extraction / Luis Felipe Müller de Oliveira Henriques ; advisor: Ruy Luiz Milidiú. – 2017.
68 f. : il. color. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.
Inclui bibliografia

1. Informática – Teses. 2. Aprendizado de máquina. 3. Processamento de linguagem natural. 4. Extração de citações. 5. Redes neurais. 6. Deep learning. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgements

First and foremost, to my advisor, Ruy, for giving me the opportunity to work with him on this project, for his faith in me, and for his valuable mentoring during this period.

To my family, in particular, my parents and my sisters, for being my greatest point of support and inspiration.

To my girlfriend, Marina, for her genuine companionship, affection and for being so understanding all this time.

To all my friends at PUC, in particular, Rafael, for sharing this not-always-easy journey.

To my colleagues at LEARN.

To PUC-Rio.

To all of you, my sincere thanks!

Abstract

Henriques, Luis Felipe Müller de Oliveira; Milidiú, Ruy Luiz. **Deep Architecture for Quotation Extraction**. Rio de Janeiro, 2017. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Quotation Extraction and Attribution is the task of identifying quotations from a given text and associating them to their authors. In this work, we present a Quotation Extraction and Attribution system for the Portuguese language. The Quotation Extraction and Attribution task has been previously approached using various techniques and for a variety of languages and datasets. Traditional models to this task consist of extracting a rich set of hand-designed features and using them to feed a shallow classifier. In this work, unlike the traditional approach, we avoid using hand-designed features using unsupervised learning techniques and deep neural networks to automatically learn relevant features to solve the task. By avoiding design features by hand, our machine learning model became easily adaptable to other languages and domains. Our model is trained and evaluated at the *GloboQuotes* corpus, and its F_1 performance metric is equal to 89.43%.

Keywords

Machine learning; Natural language processing; Quotation extraction; Neural networks; Deep learning.

Resumo

Henriques, Luis Felipe Müller de Oliveira; Milidiú, Ruy Luiz. **Arquitetura Profunda para Extração de Citações**. Rio de Janeiro, 2017. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A Extração e Atribuição de Citações é a tarefa de identificar citações de um texto e associá-las a seus autores. Neste trabalho, apresentamos um sistema de Extração e Atribuição de Citações para a língua portuguesa. A tarefa de Extração e Atribuição de Citações foi abordada anteriormente utilizando diversas técnicas e para uma variedade de linguagens e datasets. Os modelos tradicionais para a tarefa consistem em extrair manualmente um rico conjunto de atributos e usá-los para alimentar um classificador raso. Neste trabalho, ao contrário da abordagem tradicional, evitamos usar atributos projetados à mão, usando técnicas de aprendizagem não supervisionadas e redes neurais profundas para automaticamente aprender atributos relevantes para resolver a tarefa. Ao evitar a criação manual de atributos, nosso modelo de aprendizagem de máquina tornou-se facilmente adaptável a outros domínios e linguagens. Nosso modelo foi treinado e avaliado no corpus GloboQuotes e sua métrica de desempenho F_1 é igual a 89.43%.

Palavras-chave

Aprendizado de máquina; Processamento de linguagem natural; Extração de citações; Redes neurais; Deep learning;

Table of contents

List of figures

List of tables

List of abbreviations

NLP	–	*Natural Language Processing*
IE	–	*Information extraction*
ETL	–	*Entropy Guided Transformation Learning*
POS	–	*Part of Speech*
MD	–	*Mention Detection*
RE	–	*Relation Extraction*
CR	–	*Correference Resolution*
NER	–	*Named Entity Recognition*
CRF	–	*Conditional Random Fields*
DP	–	*Dependency Parsing*
QS	–	*Quotation Start*
QE	–	*Quotation End*
AD	–	*Author Distance*
NN	–	*Neural Network*
LSTM	–	*Long Short Term Memory Network*
RRN	–	*Recurrent Neural Network*
GPU	–	*Graphics Processing Unit*

Statistics is the grammar of science.

Karl Pearson.

1
Introduction

The Field of Natural Language Processing (NLP) aims to convert human language into a formal representation that computers are able to manipulate. Current final applications include information extraction, machine translation, summarization, search and human-computer interfaces (8). Quotation Extraction and Attribution is one of these applications, whose the task consists of automatically extracting quotes from a text and attributing those quotes to their authors. In Figure 1.1, we show an illustrative example of the quotation extraction and attribution task in which the quotation is underlined, and its author's name is in bold text.

> **Nélio Machado**$_1$ que defende Daniel Dantas, considerou 'estranha'$_1$ a acusação de que Dantas teria cogitado subornar o juiz. 'Isso é o fim da picada. Completamente sem fundamento e bem no dia em que o Daniel vai prestar depoimento. Estou inclinado a pedir suspeição dele (Fausto de Sanctis). Acho muito estranho, tem conteúdo de mais armação do que qualquer outra coisa'$_2$ disse **ele**$_2$.

Figure 1.1: Quotation extraction and attribution example.

News stories are often driven by the quotes made by politicians, musicians and celebrities. When these stories exit from the news cycles, the quotes they contain are often forgotten by both readers and journalists (34). A system that automatically extracts quotes and attributes those quotes to the correct author would enable readers and journalists to place news in the context of all comments made by a person on a given topic (40).

In 2008, Google released a quote aggregator platform – The Google "In-Quotes". Although it contained quotes from varied sources, it gained most of its fame from the presidential elections in the United States. This platform was very well received by the public since it provided a way to compare what each politician and thus to use this information in debates to foment discussion (41).

Besides the usefulness of such a system to politics, other areas of society may take advantage of it. For example, companies or individuals may look for quotations by themselves or by other people, and extract information regarding the success of their products or initiatives. In general, there is a great amount of meaningful information to be retrieved in these quotes.

Nowadays, most of the works on quotation extraction address to a set of subtasks that are solved by shallow machine learning algorithms applied in conjunction with a rich set of hand-designed features (41, 9, 35, 15, 40, 34, 10). The algorithms are shallow in the sense that the classifier is often linear and to achieve good performance with them we must incorporate many hand-engineered features that are also specific for the task at hand and to the language in use (7). In other words, for each subtask of the Quotation Extraction and Attribution task, the researchers themselves discover intermediate representations to the text by engineering specific features to the problem. This approach is effective since researchers can leverage a large body of linguistic and task-specific knowledge (8).

In this work, we attempt to define a machine learning model for recovering and attributing quotes to authors on the *GloboQuotes* corpus (15) while avoiding engineering features by hand. Instead, we use a single learning system able to discover adequate internal representations that are shared across all subtasks needed to solve the main task of *Quotation Extraction and Attribution*. Our intention to avoid task-specific engineered features led us to ignore a large body of linguistic knowledge (8).

In Table 1.1, we present the quality of our models assessed in the test set of the *GloboQuotes* corpus. Our model obtains a F_1 score of 89.43%, which is an error reduction of 54.43% when comparing to previous work on the same corpus (15).

Model	*Precision%*	*Recall%*	F_1*%*
This Work	**90.70**	**88.20**	**89.43**
SP_WIS	83.24	71.49	76.80
ETL	69.44	73.17	71.26
Rule Based	64.35	67.8	66.03

Table 1.1: Quotation Extraction Performance Comparison

This dissertation is structured as follows. In Chapter 2, we further describe the Quotation Extraction and Attribution task and its decomposition into subtasks. Chapter 3 describes the GloboQuotes corpus and its statistics. In Chapter 4, we present our methodology. Chapter 5, we show the results achieved in each subtask and also in the whole problem. Finally, in Chapter 6, we conclude with a short discussion on our results and future works.

2
Quotation Extraction and Attribution

Quotations are a crucial source of information, particularly, in some news articles, more than 90% of sentences can be reported speech (34).

Quotation Extraction and Attribution is the task of extracting the content span of all quotations within a given document and attributing those quotes to their authors.

In this chapter, we describe the *Quotation Extraction and Attribution* task and its tasks decomposition.

2.1
Task Decomposition

Quotation Extraction and Attribution consists of identifying quotations in a given text and associating them to their authors. We decompose this task into three subtasks:

1. Quote Identification.
2. Author candidates identification.
3. Quote Attribution.

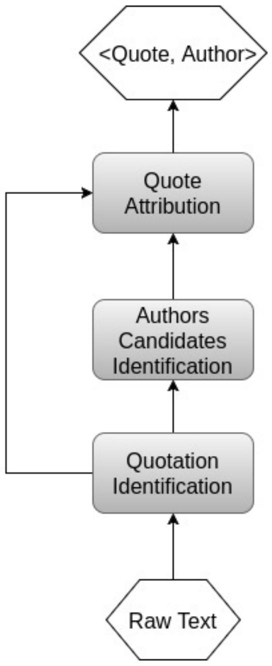


Figure 2.1: Task decomposition workflow

We present a workflow diagram of the subtasks in figure 2.1. In the remaining of this chapter we define and give illustrative examples for each subtask in the diagram.

2.2 Quote Identification

Quote Identification is the task of finding the spans that represent quotes within a document. There are three types of quotes that can appear in a running text (34):

1. *Direct quote* is a report of the exact words of an author or speaker. It appears entirely between quotation marks ("'-);

2. *Indirect quote* is a paraphrase of someone else's words. It reports on what a person said without using the exact words of the author. It is also called indirect discourse and indirect speech and does not appear between or contain quotation marks;

3. *Mixed quote* is an indirect quotation that includes a directly quoted expression. The direct portion is often just a single word or brief phrase;

As we can see in table 2.1, these three kinds of quotations are different in their construction and structure.

Type	Example
Direct quote	GLOBOESPORTE.COM: Qual a importância da conquista da Copa do Mundo de 1958?
Direct quote	"As colunas da entrada também foram atingidas", disse ela, em entrevista ao G1, por telefone.
Indirect quote	Lula ressaltou que o Brasil se consolidou como o principal parceiro da Áustria na América do Sul.
Mixed quote	O Copom disse ainda acreditar que a atual postura de "polıtica monetária" [subida dos juros], a ser mantida "enquanto for necessário", irá assegurar a convergência da inflação para a trajetória das metas.

Table 2.1: Quote types.

In Figure 2.2, we show an illustrative example of the Quote Identification subtask, in which all quotations found in the text are underlined.

Nélio Machado, que defende Daniel Dantas, considerou ’estranha’ a acusação de que Dantas teria cogitado subornar o juiz. ‘Isso é o fim da picada. Completamente sem fundamento e bem no dia em que o Daniel vai prestar depoimento. Estou inclinado a pedir suspeição dele [Fausto de Sanctis]. Acho muito estranho, tem conteúdo de mais armação do que qualquer outra coisa’ disse ele.

Figure 2.2: Illustrative example of Quote Identification subtask.

The corpus (15) that we use in this work only contains tag annotations to direct quotes and to direct portions of mixed quotes, so we limit ourselves to only detecting them.

2.3 Author Candidates Identification

The task of Author Candidates Identification can also be viewed as a first step of the *Mention Detection* (MD) task. An accurate *MD* system is vital prerequisite not only for the Quotation Extraction and Attribution task but also for a variety of Natural Language Processing tasks such as Relation Extraction (RE) and Coreference Resolution (CR) (38, 44). The mention detection task addresses to the identification and classification of entity mentions, whether named ("Fernando Henrique"), nominal (*"o presidente"*) or pronominal (*"ele"*, *"ela"*), and classifies them into some predefined types of interest such as: Person, Organization or Location (38, 44, 32).

Since our goal is to find the Author Candidates of a quote, our model only needs to be able to find *mentions* that appear in the context of the quotation that is being evaluated. In other words, we interpret any mention that appears around a quote as an author candidate to that quote.

Nélio Machado, que defende **Daniel Dantas**, considerou 'estranha' a acusação de que **Dantas** teria cogitado subornar o **juiz**.‘Isso é o fim da picada. Completamente sem fundamento e bem no dia em que o Daniel vai prestar depoimento. Estou inclinado a pedir suspeição dele [Fausto de Sanctis]. Acho muito estranho, tem conteúdo de mais armação do que qualquer outra coisa' disse **ele**.

Figure 2.3: Illustrative example of Author candidates identification subtask.

In Figure 2.3, we show, in bold text, all author candidates found in the same example from Figure 2.2. It is important to note that Author candidates can not lie inside a quotation.

2.4 Quote Attribution

Quote attribution is the task of, given a document, containing a set of quotes and a set of mentions, linking each quote in the document to the mention that represents its author's entity (34).

Nélio Machado$_1$, que defende **Daniel Dantas**, considerou ‘estranha’$_1$ a acusação de que **Dantas** teria cogitado subornar o **juiz**. ‘Isso é o fim da picada. Completamente sem fundamento e bem no dia em que o Daniel vai prestar depoimento. Estou inclinado a pedir suspeição dele [Fausto de Sanctis]. Acho muito estranho, tem conteúdo de mais armação do que qualquer outra coisa’$_2$ disse **ele**$_2$.

Figure 2.4: Illustrative example of Quote Attribution subtask.

Figure 2.4 shows an illustrative example of two associations between quotation and authors. The link between them is represented by the subscripted tags in the text, that is, quotes and authors are tagged with the same subscript tag.

2.5 Related Works

Quotation Extraction and Attribution has been previously approached using different techniques and for several languages. Previous work on Quotation Extraction and Attribution includes both rulebased and machine learning approaches.

The work of Mamede and Cholera, in 2004, is one of the earliest contributions in this area. It is developed on top of a created corpus, based on a set of narrative text (28). The objective is to differentiate narrative text in the indirect speech from quoted text and use this information for automatic storytelling. Besides addressing the author identification and the quotation attribution, they created an algorithm for quotation detection based on a set of twelve heuristics regarding the lexical form of the text to which a predefined degree of trust is applied. The proposed solution to quotation attribution uses a set of rules that look in the indirect discourse surrounding a span of direct speech for lexical characteristics and verbs, which suggests the true author of a quoted speech. A decision tree is then trained to identify the rule that can predict the correct character for a given direct speech sample. The testing corpus consists of 35 samples, and the reported score is $65,7\%$ of accuracy.

In 2007, Pouliquen and Steinberger introduced an automatic online system to detect quotations and attribute them to the proper author in Multilanguage news text (40). The system uses lists of:

- verbs of speech(e.g. said, commented, "disse","dice").
- quotation marks(e.g. ’,",-).
- general modifiers(e.g. yesterday).
- determiners(e.g. the).
- people’s names.

With the objective of generalizing the model for 32 languages, the linguistic input is kept as simple as possible. The method developed relies on regular expressions to identify quotations and to associate them to their authors. The recall of the system, reported in this work, is 54%, and the reported precision is 81.7%.

In 2009, a system called *Verbatim* (10) is created trying to solve the problem of Quotation Extraction and Attribution and the problem of classifying news topics. Similarly to work in (40), the *Verbatim* system uses 19 regular expressions and a list of verbs of speech to identify quotations and their authors.

A slightly different approach, proposed by Elson and McKeown in 2010 (14), addresses the problem of Quotation Attribution in literary narratives, adding a new step to the task, in which the quotes are classified into one of a predefined set of seven categories. According to the authors, this step allows the clustering of scenarios where the syntax strongly implies a particular solution. The resulting classes are then sent to a specifically trained model, for the syntax, that calculates a score for the author. Then, the author with the highest score is assigned to the quote. This system uses coreference resolution information, POS tagging, a list of verbs of speech and a set of hand designed features that are used to build a machine learning model and also in the preprocessing steps. It is reported that this proposed approach achieves an average accuracy of 83% in a corpus containing about 3000 quotes removed from literary narratives (e.g. *Pride and Prejudice*).

A more recent work on Quotation Extraction and Attribution, presented in 2012 by Fernandes (15), is also based in the Portuguese language. His work introduced the *GloboQuotes* corpus, which is used to train a machine learning model to solve the task of Quotation Extraction. In his, work Fernandes compared the performance of a model using the Entropy Guided Transformation Learning (ETL) algorithm (12) with a model using Structured Perceptron model (6). In those models, coreference, POS and NER tags are used to gather features to feed the algorithms. The reported performance for the whole task is a F_1 score of 76.8%.

The work by Pareti et al. (34, 35) also uses Machine Learning algorithms approaching the task as a sequence labeling task. They use three sequence decoding models: Greedy, Viterbi and Conditional Random Fields (CRF). The Quotation Extraction is performed using regular expressions to recover the text between quotation marks. In the model evaluation, an attribution is considered correct if a quote is assigned to the right coreference chain. As a preprocessing step, they removed adjectives, adverbs and other POS tagged words that were considered as not to containing relevant information. They applied their approach to two datasets, and the reported results were of 84.1% and 91.2% of accuracy.

In 2017, dos Reis Silva (11) developed a quotation extractor system that approaches the problem of Quotation Extraction and Attribution on direct and indirect quotations. His work focuses on Portuguese language and delivers a new corpus containing golden annotations for indirect quotations as well. As in Fernandes's work, dos Reis Silva uses a Structured Perceptron model to solve the problem of quotation attribution. His method is based on rules that rely on the Dependency Parsing (DP) trees and a list of verbs of speech, to find the quotations and their author candidates. His system also uses POS and NER tags to gather features to the Structured Perceptron. The reported performance is F_1 of 66%.

Our proposal differs from previous work in many senses. To the best of our knowledge, this is the first work on Quotation Extraction and Attribution that uses a unified deep neural network architecture to solve the task. With the emerging technologies of deep learning, many works on how to deliver prior knowledge from untagged data have been published in the literature. This work also uses an unsupervised learning technique to deliver knowledge from a large unlabeled dataset, which is composed by the junction of tree corpus. Another peculiar characteristic that differs our work from previous works mainly in the quotation attribution is the fact that we have modeled this task as closed domain classification task, in which we have the same predefined set of labels for each quote example. Also, since all information that we use to solve the whole task is contained in the text and we do not make hard features engineering, it is easier to adapt our model to other languages and domains. The previously proposed systems would probably need adaptations or work on feature re-engineering to correctly classify quotations which would be time-consuming (7).

3
GloboQuotes Corpus

GloboQuotes is a corpus built by Fernandes (15) at the Learn Lab of the Pontifícia Universidade Católica do Rio de Janeiro. When this corpus was created, there wasn't any public available quotation corpus for the Portuguese language. The Corpus is based on news pieces in Portuguese from the globo.com portal and contains the following Golden annotations for:

- Named Entities
- Quotation bounds and composition
- Association between quotations and authors
- A selected set of coreferences
- Clause

Besides, the corpus provides annotations for constituency parsing and POS tagging, both obtained using a state of art tagger (33).

The Corpus is codified on a per token basis, that is, each token (*word*) is tagged with the same set of possible annotations. In this chapter, we present some significant corpus statistics and concepts that are useful to understand the Quotation Extraction and Attribution task.

3.1
Corpus Composition

GloboQuotes was originally generated by sampling news from a bigger untagged corpus, called *Globocom*, containing more than 44,000 pieces of news dated from August 2007 to August 2008. The news in this raw corpus can be divided into ten categories of news, and its distribution is presented in Figure 3.1.

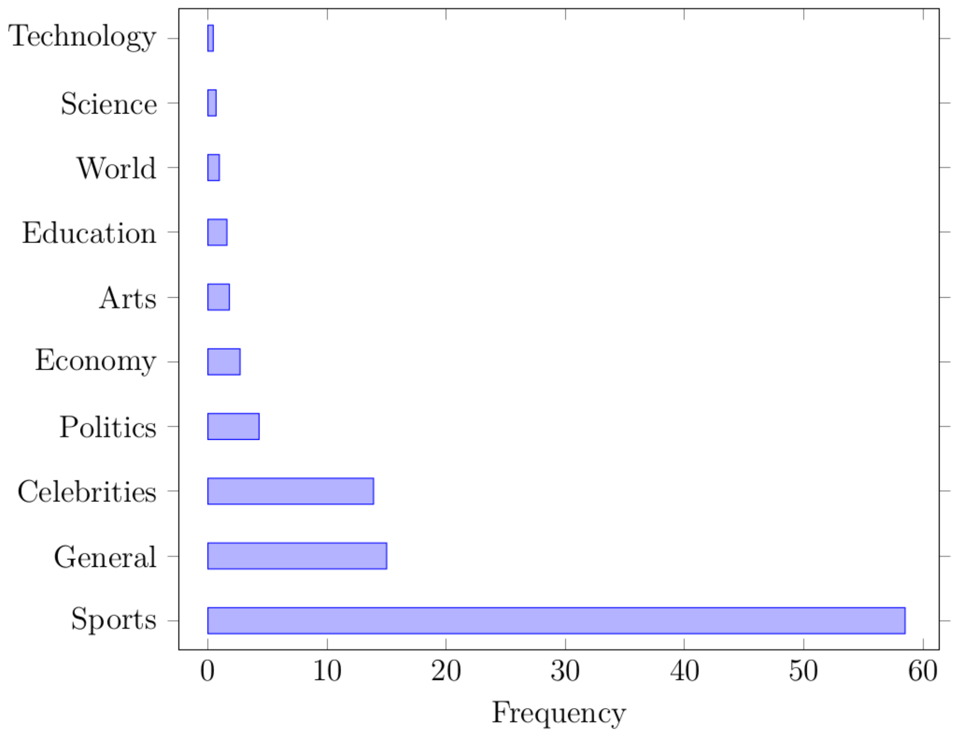


Figure 3.1: Distribution of corpus categories.

GloboQuotes preserves this distribution.

3.2 Document

GloboQuotes is divided into two groups:

– *Training set* : composed by 552 documents.

– *Testing set*: composed by 133 documents.

The training set should be used to train and validate models for each subtask of the Quotation Extraction and Attribution task. On the other hand, the testing set should be used to evaluate the trained models according to some evaluation metric (e.g. accuracy).

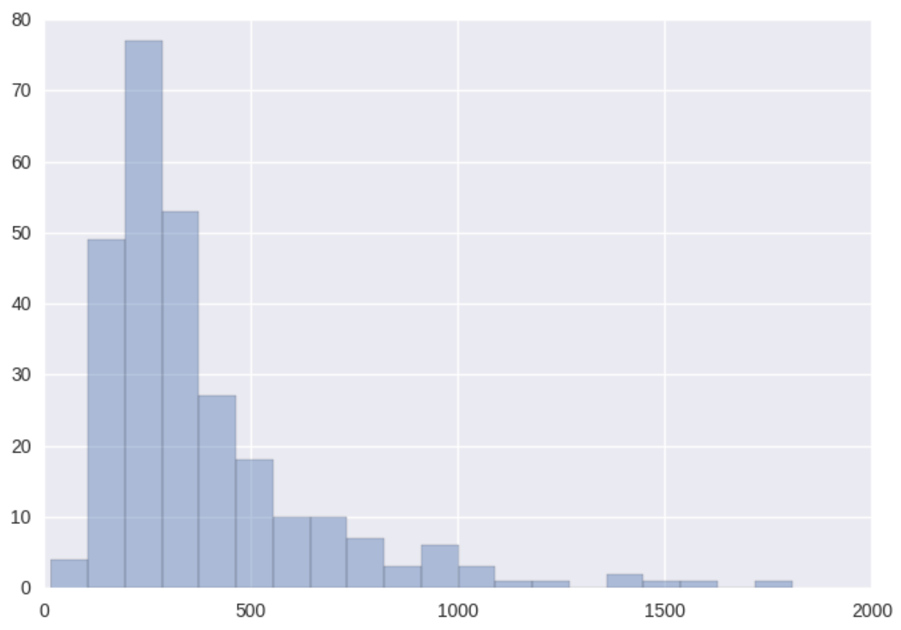


Figure 3.2: Document Length Distribution.

Documents have different lengths, varying from 16 tokens to 1807 tokens in a given document of the training set group. The mean case is approximately 379 tokens per document. Figure 3.2 shows a histogram with length distribution across the training set.

3.3 Quotation

Quotations are the heart of the task. They contain the information we want to extract and, as we will see later in this chapter, their position and the words that surround them bring us tips about their authors (34).

In *GloboQuotes*, quotations are annotated using IOB format (42). In the example of Table 3.1, we have separated the features into three different attributes. In the quote start (QS) feature, we use S to mark the quotation starting token. Similarly, we use E in the quote end (QE) feature to mark the quotation end token. In the Quote feature, we assign the q tag to each token that belongs to a quote.

In its training set, *GloboQuotes* presents 764 annotated quotations distributed across 274 news documents. The number of quotes per document ranges from 1 to 25 quotations in a single document, being the mean case approximately 3 quotes per document and the most common case is documents containing 2 quotes in 91 examples.

Quotations have different lengths, varying from 2 tokens to 161 tokens. The mean case is approximately 36 tokens per quotation, being the most common example, quotes containing 36 tokens in 22 cases.

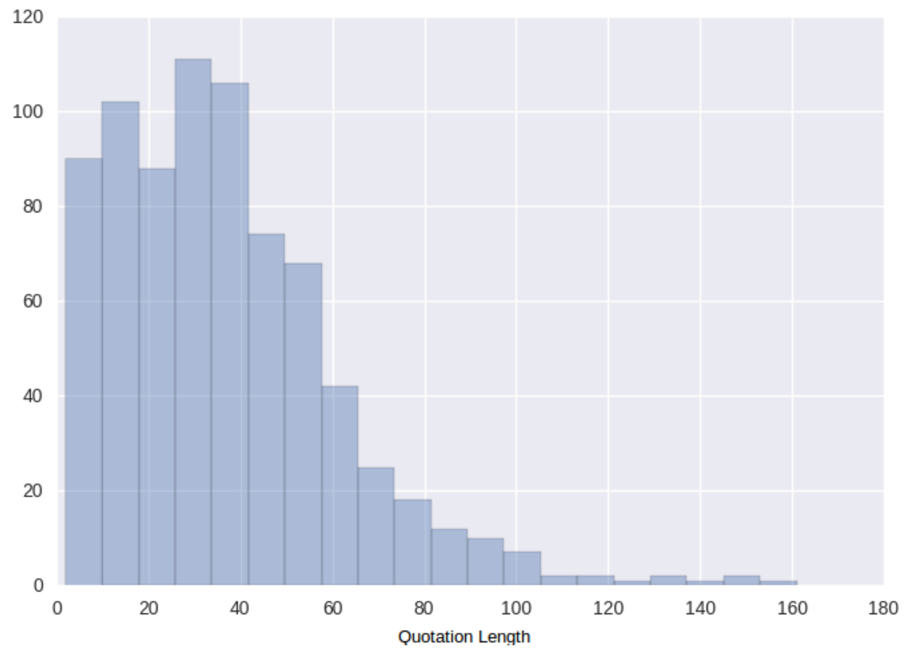


Figure 3.3: Corpus Quotation Length Distribution.

Figure 3.3 shows the length distribution across the training set.

3.4 Author of a Quote

One of the most important annotations in *GloboQuotes* is the annotation that links authors to their quotes. This annotation is represented by the relative distance of a quote to its author. The relative distance is counted by the number of golden annotations of coreferences seen between a quote and its authors, with the counting starting from the quote. In this tag, a positive signal (+) before the distance to the author is used to indicate whether the author is on the right side of the quotation, that is, after the end of the quote. In the same way, a negative signal (-) before the distance is used to indicate whether the author is on the left side of the quotation, that is, before the quotation begin.

Word	**Temer**	**ataca**	**gestão**	**de**	**Dilma**	**e**	**alega**	**"**	**havia**	**déficit**	**de**	**verdade**	**no**	**governo**	**"**
Coref	ref00	-	-	-	ref01	-	-	-	-	-	-	-	-	-	-
QS	-	-	-	-	-	-	-	-	S	-	-	-	-	-	-
QE	-	-	-	-	-	-	-	-	-	-	-	-	-	E	-
Quote	-	-	-	-	-	-	-	-	q	q	q	q	q	q	-
AD	-	-	-	-	-	-	-	-	-2	-2	-2	-2	-2	-2	-

Table 3.1: Author of a quote tag example.

Table 3.1 shows an illustrative example of how the relative distance works. In the table, the author column is colored in gray. The rows Coref, QS, QE, Quote, and AD are, in the respective order, representing marks of Coreference, Quotation Start, Quotation End, Quotation and Author Distance. By now, we can consider any token that is tagged with any tag different of - as a *mention*.

Figure 3.4 shows this distance distribution across the training set. As we can see, the four predominant relative distances are "-3", "-2", "-1", and "+1", whose respective frequencies are 3.64%, 13.49%, 35.05% and 46.06%. Other cases, together, account for 1.78% of the training set.

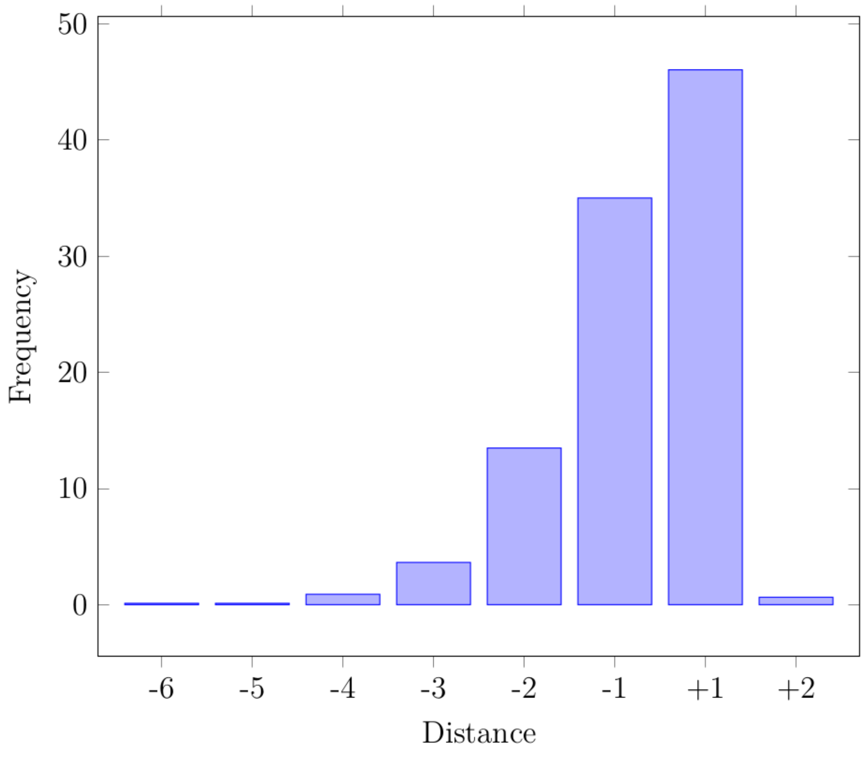


Figure 3.4: Author distance distribution.

Another interesting point is that the total number of quotations found in the training set is 764, but the count of mentions marked as an author of any quotation is just 700, which indicates that, even though in a few cases, one mention can be the author of more than one quotation.

3.5

Mention

Mentions are references in the text to real world entities along with their types, e.g. people, organizations and others (2, 32, 49, 44). References to real world entities can be either named (e.g. "Donald Trump"), nominal (e.g."the president") or pronominal (e.g. "he", "she") (49).

GloboQuotes does not have golden annotations for all mentions that are found in its documents. Instead, the corpus provides golden annotations for a selected subset of coreference resolution chains, in which it is guaranteed that this subset contains all Quotes Authors and their competing Authors candidates chains. By now, we consider a coreference chain the set of mentions in a given text that points to the same entity, e.g. a person or organization.

In this work, we consider as a mention any token or sequence of tokens that are marked as being part of any coreference chain.

In the training set, a total of 3,509 annotated mentions heads (the first token in a mention) is counted, in which, 72.95% of the mentions are named entities, 20.91% are nominal mentions and 6.14% are pronominal mentions.

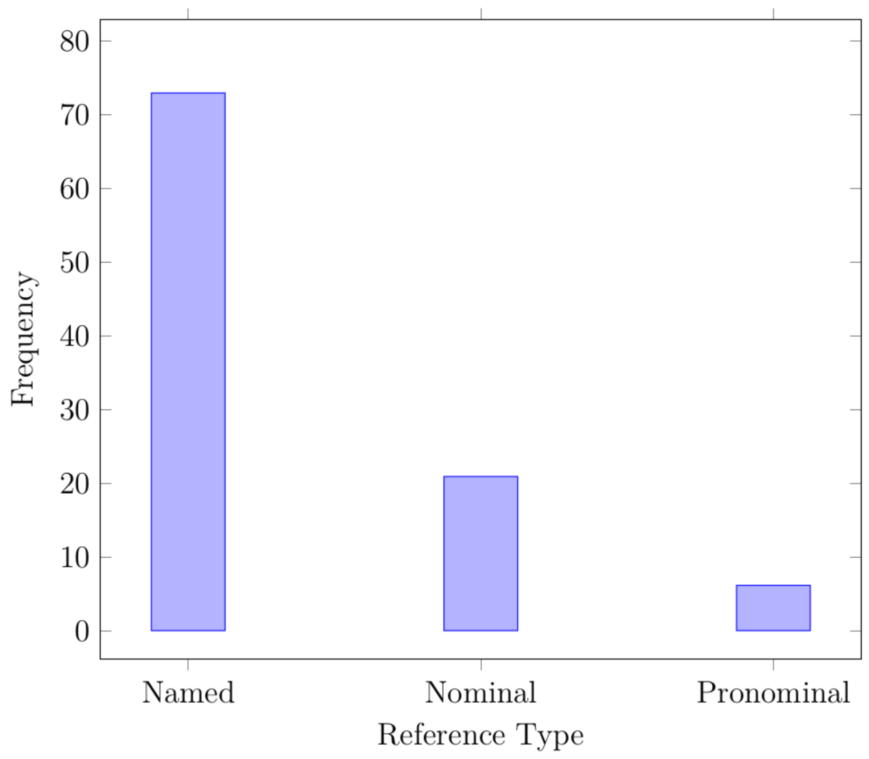


Figure 3.5: Distribution of mention reference types.

Figure 3.5 presents the distribution of mentions according to their pointing types.

4
Methodology

The approach proposed by this work is to handle the problem as a set of supervised classification problems. Classification is the task of assigning a correct label to a given input.

In this work, we use the *GloboQuotes* corpus to learn classification functions. A generic classification function h determines a mapping from data to its respective label.

We denote the input set by X, composed by a set of x_i vectors and the output classes (or labels) as a finite set of $Y = \{c_1, ..., c_k\}$. We also refer to $x \in X$ and $y \in Y$ as particular instances for X and Y respectively.

In the case of Quote Attribution subtask, the input can be seen as a set of quotes recovered from a text and the classes as the candidates of authors to which they might belong. That is, vectors x_i contain representations of quotes and their candidates of authors and y labels uniquely identify the quotations' authors.

The learned classification function $h : X \rightarrow Y$ can be applied on an unseen input x to get an estimated $\hat{y}$ instance.

$$\hat{y} = max_k(h(x)) \tag{4-1}$$

In this chapter, we present the methodology and models used to approach the Quotation Extraction and Attribution task.

4.1
Neural Networks

The task of Quotation Extraction and attribution (and all its subtasks) is commonly approached by extracting from a text a comprehensive set of hand-designed features which are then fed to a standard classification algorithm, that is often linear (41, 9, 35, 15, 40, 34, 10). The choice of features is a completely empirical process, mainly based on linguistic intuition and trial and error (8).

With the objective of addressing the Quotation Extraction and Attribution, while avoiding hand engineered features, we propose a unified neural network architecture with multiple layers that are used in the whole task.

A multilayer neural network $h_\theta(x)$, with parameters θ and L layers, can be seen as a composition of functions $h_\theta^L(.)$:

$$h_\theta(x) = h_\theta^L(h_\theta^{L-1}(....h_\theta^1(x))) \tag{4-2}$$

Our neural network takes the input and processes it by several layers of automatic feature extraction (7). The features computed in deep layers are automatically trained by back propagation (26) to be relevant to the task (8).

Our architecture is summarized in figure 4.1. The first layer extracts features for each token in the input. The second layer mixes each token representation in a linear sequence order. This layer is responsible for representing how the sequential text information affects each word in the sequence (31, 19). The subsequent layers extract features from a window of tokens to get its local structure (8). Finally, the last layer is a simple *SoftMax* layer that is responsible for calculating a score to each possible output.

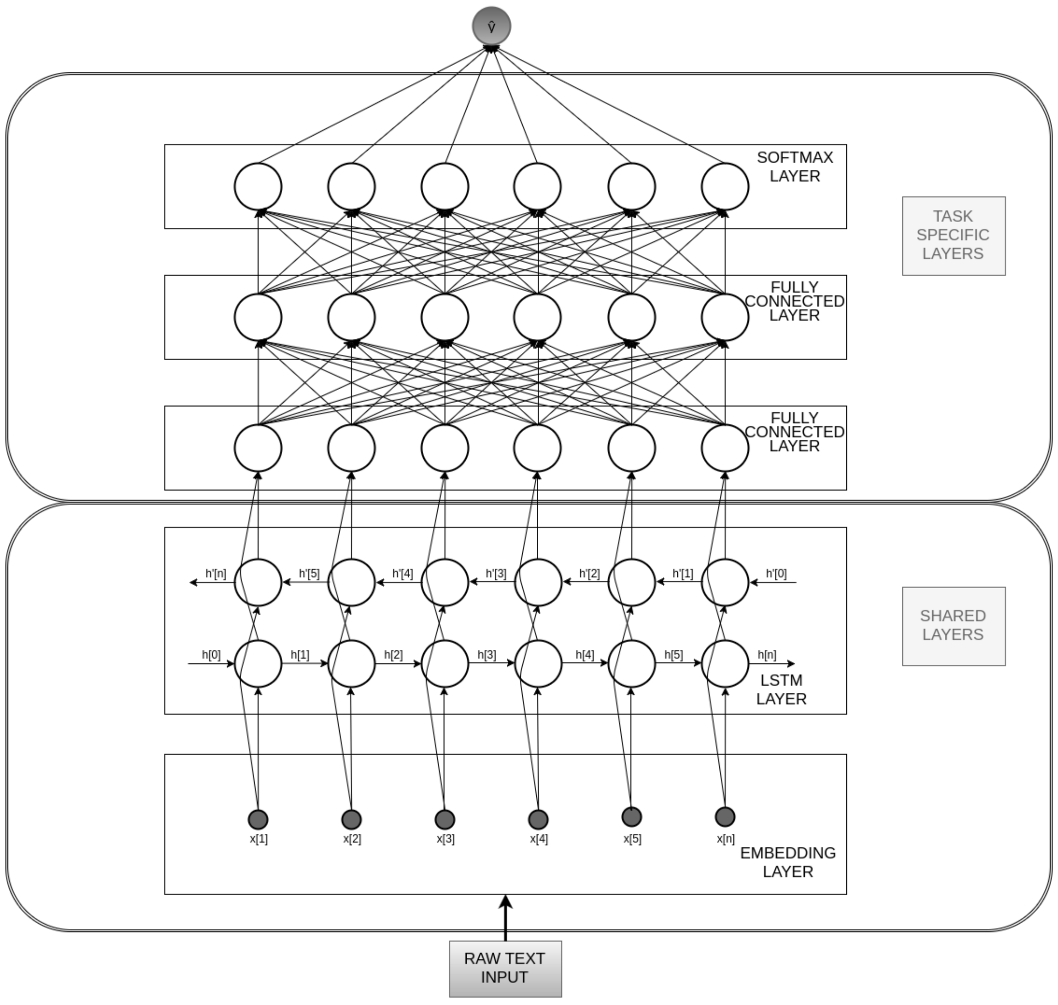


Figure 4.1: Deep Architecture.

The two first layers share their parameters along all subtasks. This linkage can be seen as a soft constraint regularization in the joint cost function that biases the models towards common representations (3). In this way, the layers of our Neural Network can be divided in two types:

1. Specific Layers; are trained only by examples of their subtasks. These are upper layers of the neural network in the Figure 4.1.
2. Shared Layers; are shared across all subtasks and are trained by the data of all subtasks. These are lower layers of the neural network in Figure 4.1.

4.2 From Tokens to Feature Vectors

A key point of our architecture is its ability to perform well with the use of raw words. The ability for our method to learn good word representations is thus crucial to our approach. For efficiency, words are fed to our architecture as indexes taken from a finite dictionary D. Obviously, a simple index does not carry much useful information about a word (7). However, the first layer of our network maps each of these word indexes into a feature vector, by applying a *lookup table* operation. So, given the raw text, a relevant representation of each word is then given by the *lookup table*. Formally, each word $i \in D$ is *embedded* into a d-dimensional space using a lookup table $LT_W(.)$:

$$LT_W(i) \rightarrow W_i \tag{4-3}$$

where $W \in \mathbb{R}^{|D| \times d}$ is a matrix of first layers parameters to be learnt, $W_i \in \mathbb{R}^d$ is the i^{th} row of W and d is the word vector size. Given a sequence of T tokens $[i]_1^T$ in D, the sequence is thus transformed into a series of vectors $\{W_1, W_2, W_3, W_4, ..., W_T\}$ by applying the lookup table to each of its tokens. Figure 4.2 shows an illustrative example of how this layer works.

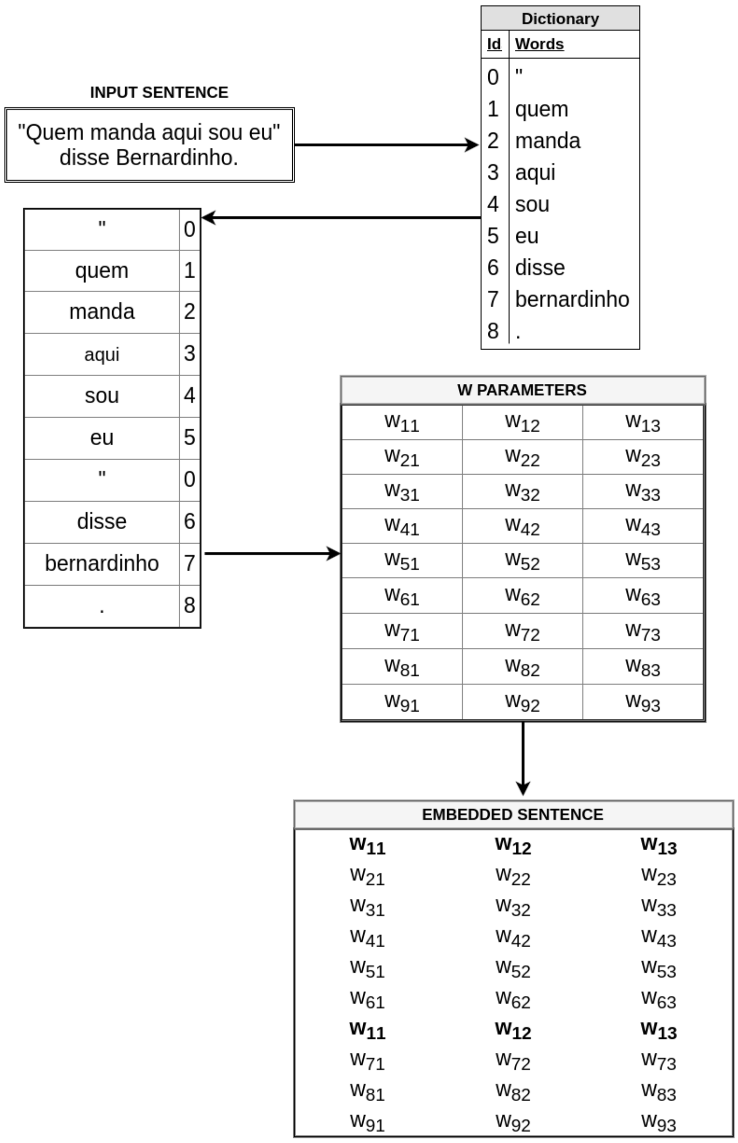


Figure 4.2: Lookup table operation example diagram.

Mathematically speaking, this first layer defines a linear operation. Each token index represents a *one-hot encoded*[1] vector $x \in \mathbb{R}^{|D|}$, and $W \in \mathbb{R}^{|D| \times d}$ is a matrix of the layers parameters (29). In this definition, the output of the layer is given by:

$$LT_W(i) = x^T W \tag{4-4}$$

which is, essentially, a copy of the i^{th} row of the matrix W. Thus, the example presented in Figure 4.2 is translated to this definition as follows:

[1]One-hot encoded vector is a vector with all its positions assigned to zero except for the index position that is assigned to 1.

$$X^T = \begin{bmatrix} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{bmatrix} ; W = \begin{bmatrix} W_{11}&W_{12}&W_{13}\\ W_{21}&W_{22}&W_{23}\\ W_{31}&W_{32}&W_{33}\\ W_{41}&W_{42}&W_{43}\\ W_{51}&W_{52}&W_{53}\\ W_{61}&W_{62}&W_{63}\\ W_{71}&W_{72}&W_{73}\\ W_{81}&W_{82}&W_{83}\\ W_{91}&W_{92}&W_{93} \end{bmatrix} \quad (4\text{-}5)$$

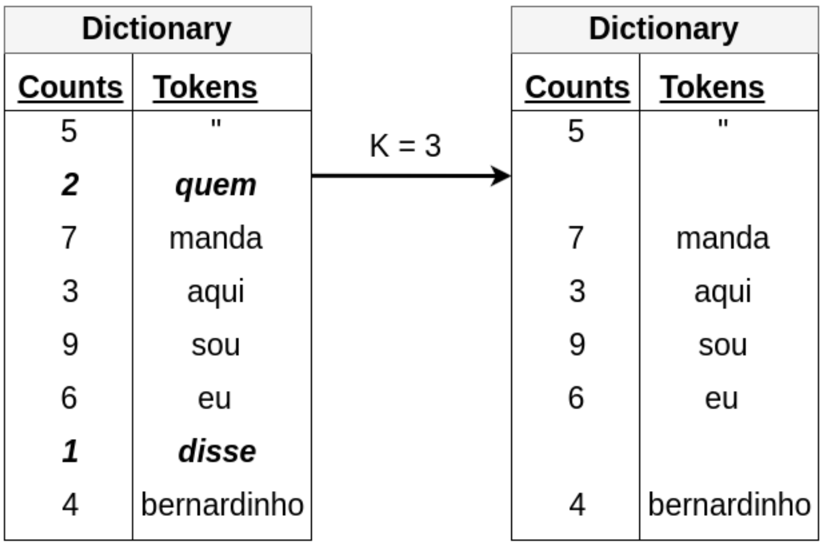
$$h^1_\theta(X) = X^T W = \begin{bmatrix} W_{11}&W_{12}&W_{13}\\ W_{21}&W_{22}&W_{23}\\ W_{31}&W_{32}&W_{33}\\ W_{41}&W_{42}&W_{43}\\ W_{51}&W_{52}&W_{53}\\ W_{61}&W_{62}&W_{63}\\ W_{11}&W_{12}&W_{13}\\ W_{71}&W_{72}&W_{73}\\ W_{81}&W_{82}&W_{83}\\ W_{91}&W_{92}&W_{93} \end{bmatrix} \quad (4\text{-}6)$$

The output of this layer is commonly called *word embeddings* (29, 30, 43, 39, 22).

4.2.1 Building the Dictionary of Tokens

The dictionary of tokens is responsible for mapping words to index numbers. To achieve this objective, the inputs are preprocessed converting all words to lowercase. Additionally, numbers, dates, email addresses and website URLs are translated into special tokens, respectively, "#number", "#date", "#email", "#url".

To avoid dealing with rare tokens, an hyperparameter "k" that specifies a minimum count for each token is added to the model. More clearly, while creating the dictionary, the occurrence of each token is computed and for each token, if its count is smaller than "k", then this token is deleted from the dictionary. Figure 4.3 shows an illustrative example of the usage of the hyperparameter "k". The example presents a dictionary of tokens (and their counts), and the hyperparameter "k" is set equal to 3. The resulting dictionary is presented in the right side of the arrow.



Dictionary	
Counts	**Tokens**
5	"
2	***quem***
7	manda
3	aqui
9	sou
6	eu
1	***disse***
4	bernardinho

Dictionary	
Counts	**Tokens**
5	"
7	manda
3	aqui
9	sou
6	eu
4	bernardinho

Figure 4.3: Lookup table operation example diagram.

Furthermore, a special token is added to the dictionary with the objective of handling unseen or deleted tokens. That is every time that our model reaches a token that is not in the dictionary, then this token is mapped to a special token that represents unseen tokens. In this work we will refer to this special token as "#UNK".

4.2.2 Extending Tokens Representations

As mentioned before, in section 4.2.1, the input tokens are converted to lower case. To avoid losing of information, we represent the capitalization as separated discrete feature:

1. All in upper case
2. First letter in upper case
3. Last letter in upper case
4. All in lower case
5. Other combinations

To add this feature f to the model, we define a new lookup table operation $LT_F(.)$ to this feature. Exactly in the same way as the lookup table described above, the output of $LT_F(.)$ is an embedding to the capitalization feature. The token representation is then a concatenation of word embedding and the capitalization feature embedding, more precisely $[LT_W(i); LT_F(f)]$.

This method can be used to extend a token representation with any discrete feature. Generally speaking, a word can be represented by n discrete features $w \in D^1 \times D^2 \times ...... \times D^n$, where D^n is a dictionary for the n^{th} feature. We associate to each feature a lookup table $LT_{F^n}(.)$ with parameter $F^n \in \mathbb{R}^{d^n \times |D^n|}$ where $d^n \in \mathbb{N}$ is the n^{th} embedding vector size. The token representation is then obtained by concatenating all lookup tables outputs (8):

$$x = [LT_{F^1}(.); LT_{F^2}(.); ......; LT_{F^n}(.)] \tag{4-7}$$

As discussed before, in section 4.2, these embeddings layers are mathematically defined as linear projection layers of feature extraction and, therefore, their results can be concatenated to any layer inputs of our neural network (31, 24, 4).

4.2.3 Leveraging Unlabeled Data

Labeling a data set can be an expensive task, especially in NLP where labeling often requires skilled linguistics. On the other hand, unlabeled data is abundant and freely available on the web. Leveraging unlabeled data in NLP tasks seems to be very attractive (7).

For this propose, our model is pre-trained using an *unsupervised neural network* on a junction of three unlabeled corpora:

1. *Globocom* corpus

2. *WikiNews* corpus

3. *WikiBooks* corpus

Globocom corpus is composed of a set of $44,000$ news dated from August/2007 to August/2008, from globo.com[2] portal. The *WikiNews* corpus consists of the set of news published in Portuguese, in 2015, on the Wikinews[3] website. In its turn, the *WikiBooks* corpus consists of the set of books present on the WikiBooks[4] website until 2015. The junction of these three corpus results in more than $91,582$ megabytes of text information, counting more than $16,641,447$ tokens.

[2]http://www.globo.com/
[3]https://pt.wikinews.org/
[4]https://pt.wikibooks.org/

Skip-gram Model

As mentioned before, our model is pre-trained with a neural network for learning distributed representations of tokens. Recently, neural networks based approaches in which words are "embedded" into a low dimensional space were proposed by various authors (29, 30, 43, 39, 22, 45, 8).

In this work, our model is pre-trained with the Skip-gram model, proposed by Mikolov (30, 29). The Skip-Gram model tries to learn tokens representations that are useful for predicting the surrounding tokens in a sequence or a document. More precisely, it uses each token as an input to a classifier with continuous projection layer and predicts tokens within a certain range before and after the current token. Since the more distant words are usually less related to the current token than those close to it, the model gives less weight to the distant words by sampling less from those tokens during the training step. Figure 4.4 shows a representative illustration for the Skip-gram model.

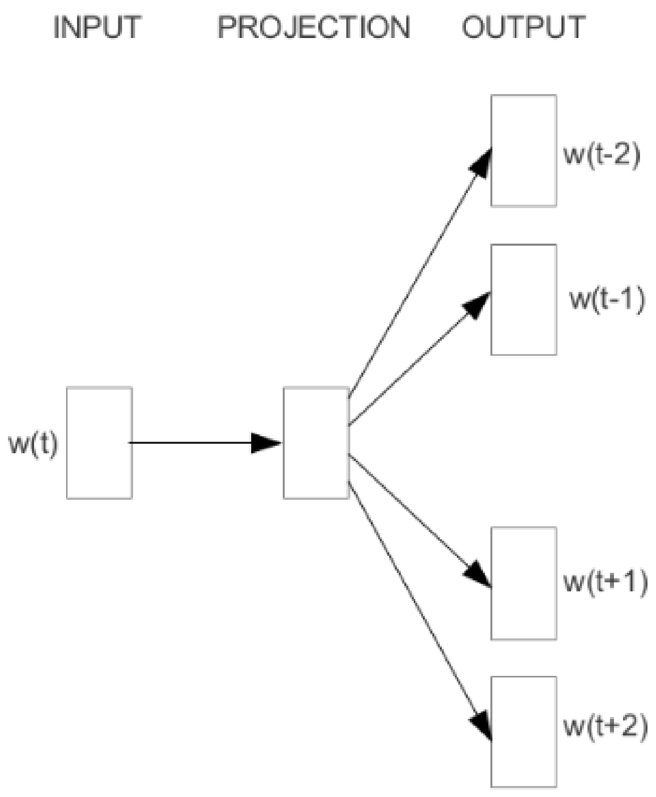


Figure 4.4: The Skip-gram model architecture

Skip-gram is an unsupervised model, and, in this work, it is trained using negative sampling technique (30). Besides during training, each token is subsampled according to the following distribution:

$$P(W_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{4-8}$$

where $f(w_i)$ is the frequency of word w_i and t is a chosen threshold typically around 10^{-5}. This formula aggressively subsamples words whose frequency is greater than t while preserving the ranking of the frequencies. The idea behind this technique is that the vector representation of the most frequent words does not change significantly after training on several examples (30).

4.3 Sequential Text Information Layers

The second layer of our neural network architecture is a Long Short Term Memory Network (LSTM) that extracts information (features) from sequences of words using the representations received from the embedding layer.

LSTMs are variants of recurrent neural networks (RNNs) designed to cope with the vanishing gradient problem inherent in RNNs (21, 16). The RNNs read a vector x_t at each time step and compute a new hidden state h_t by applying a linear map to the concatenation of previous time step's state h_{t-1} and the input, passing this through a logistic sigmoid non-linearity. Although RNNs can, in principle, model long-range dependencies, training them is difficult in practice since the repeated application of a squashing non-linearity at each step results in an exponential decay in the error signal through time (37). LSTMs address this with an extra memory cell that is constructed as a linear combination of the previous state and signal from the input.

The LSTM unit process inputs at t^{th} time step with a set of n-dimentional vectors:

1. Input gate (i_t)
2. Output gate (o_t)
3. Forget gate (f_t)
4. A Memory Cell (c_t)
5. A Hidden State (h_t)

The unit receives an n-dimensional input vector x_t, the previous hidden state h_{t-1}, and the previous memory cell c_{t-1}, and calculates the new vectors using the following equations:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^t) \tag{4-9}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{4-10}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{4-11}$$

$$u_t = tanh(W^u x_t + U^o h_{t-1} + b^u) \tag{4-12}$$

$$c_t = i_t * u_t + f_t * c_{t-1} \tag{4-13}$$

$$h_t = o_t * tanh(c_t) \tag{4-14}$$

where σ denotes to the element-wise sigmoid function, $*$ denotes to the element-wise multiplication operation, W and U are weight matrices, and b are bias vectors.

To improve the representational capacity, LSTMs can be stacked in layers (36, 13). In these architectures, the input at a higher layer, at time t, is the value of h_t computed by the lower layer. In this work, we use this kind of stack in the Author Candidates Identification subtask.

Bi-Directional LSTM

Our LSTM layer, represented in Figure 4.1, is actually a Bi-Directional layer meaning that the layer performs two passes to encode the sequences. The procedure for a sequence $X = x_1, x_2, ..., x_n$ is:

1. Forward propagation: Run the first LSTM layer from left to right over $x_1, x_2, ..., x_n$ to obtain the first hidden state vector sequence.

2. Backwards propagation: Run the second LSTM from right to left over $x_1, x_2, ..., x_n$ to obtain the second hidden state vector sequence.

3. Concatenate the output sequences of the Forward and Backwards steps.

Conceptually, the Forward and Backwards passes can be performed by different layers, with it owns parameters θ, but for consistency and simplicity, we assume that we only have a single layer that shares its parameters θ in both passes (47).

The observation at step t, the forward hidden state vector h_t^f carries information of the past word context, while the backward hidden state vector h_t^b is the summary for the future word context. Consequently, the concatenated vector $\alpha = [h_t^f; h_t^b]$ constitutes a distributed representation that is specific to the word at time step t, but still encapsulates the context information at the same time (17, 18).

4.4 Token Window Approach

Our Token Window approach assumes that the tag of token depends mainly on its neighboring words (8). That is, given a token, we consider a fixed size c of tokens around this token. As discussed in section 4.1, before each token window is processed as an input example, each token in the window is passed by several layers of feature extraction. These many layers of feature extraction produce a matrix of tokens features of fixed size $d \times c$, where d is an hyper-parameter of the model indicating the token representation size. This matrix $M \in \mathbb{R}^{d \times c}$ is transformed into a fixed size feature vector x_i. This fixed size vector x_i is used to feed one or several standard neural network layers which perform several transformations over their inputs (20). Finally, in the last layer of our neural networks, those transformed features are used to calculate a score to each possible label, putting it in a vector $\overline{y} \in \mathbb{R}^k$. Then, a classification is made by selecting the label with maximum score $\hat{y} = max_k(\overline{y})$.

For the purpose of transforming a matrix $M \in \mathbb{R}^{d \times c}$ of tokens representations in the selected window into a fixed size vector, two approaches are used in this work.

1. The flattening strategy.

2. The max strategy.

The flattening strategy consists of concatenating each row of the matrix M resulting in a vector of size $d \times c$. More formally, the resulting feature vector can be written as:

$$x = [M_1; M_2; .....; M_i; M_{i+1}; .....; M_{2c}; M_{2c+1}] \tag{4-15}$$

where M_i is the i^{th} row of the matrix M and c is the fixed number of tokens that surround the main token in the middle of the matrix M.

As well as in the flattening strategy, the max layer transforms a matrix of token windows into a fixed size vector, but instead of producing a vector of size $d \times k$, it produces a matrix of size d. In the same way that traditional convolutional neural networks usually apply a max-pooling operation, in this strategy, an element-wise max operation is applied through the columns of the matrix M. The idea of this approach is forcing the network to capture the most useful local features produced by lower layers. Additionally, the max operation downgrades the representation dimensionality of the model and provides additional robustness to positions of the most relevant features (8). Figure 4.5 illustrates the max layer operations.

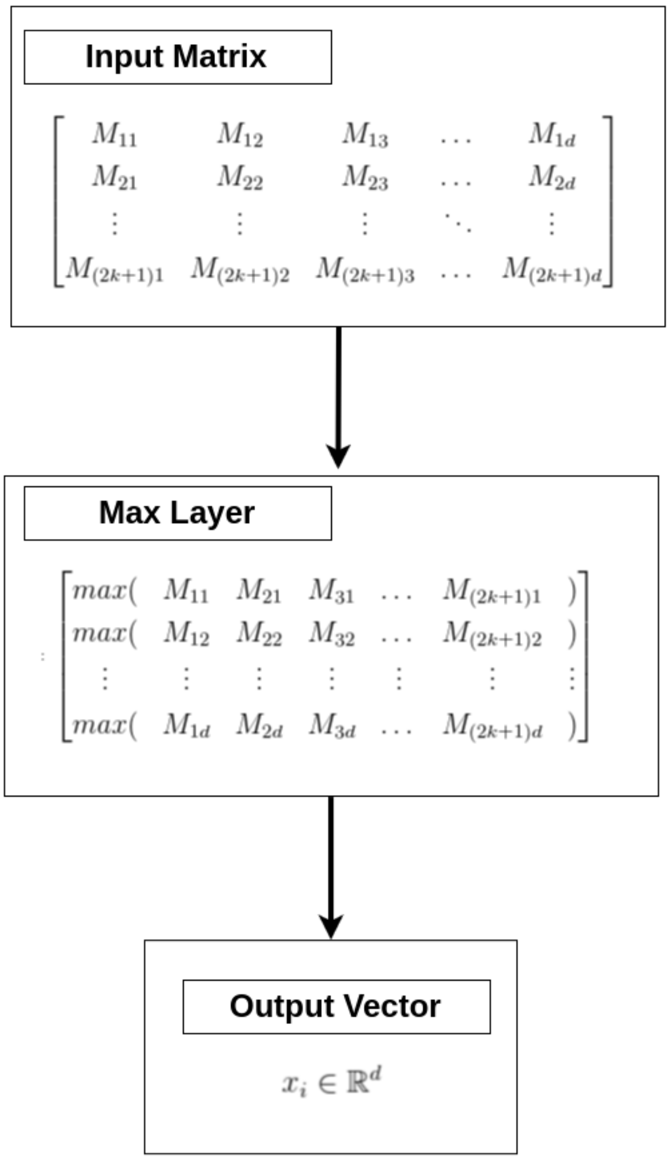

Input Matrix
$\begin{bmatrix} M_{11} & M_{12} & M_{13} & \dots & M_{1d} \\ M_{21} & M_{22} & M_{23} & \dots & M_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{(2k+1)1} & M_{(2k+1)2} & M_{(2k+1)3} & \dots & M_{(2k+1)d} \end{bmatrix}$
Max Layer
$\begin{bmatrix} max(& M_{11} & M_{21} & M_{31} & \dots & M_{(2k+1)1} &) \\ max(& M_{12} & M_{22} & M_{32} & \dots & M_{(2k+1)2} &) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ max(& M_{1d} & M_{2d} & M_{3d} & \dots & M_{(2k+1)d} &) \end{bmatrix}$
Output Vector
$x_i \in \mathbb{R}^d$

Figure 4.5: Max layer representation.

4.4.1 Encoding Details

Our token window approach is applied across all subtasks of the Quotation Extraction and Attribution. For this purpose, an additional work on encoding the inputs of our models is necessary. This extra work is done by contracting sequences of tokens into a special new token that represents this sequence in a unified way. Figure 4.6 shows an illustrative example of token contraction strategy. In this example, the original quotation size is equal to 5 tokens and it is contracted to a special single token (*#quote*).

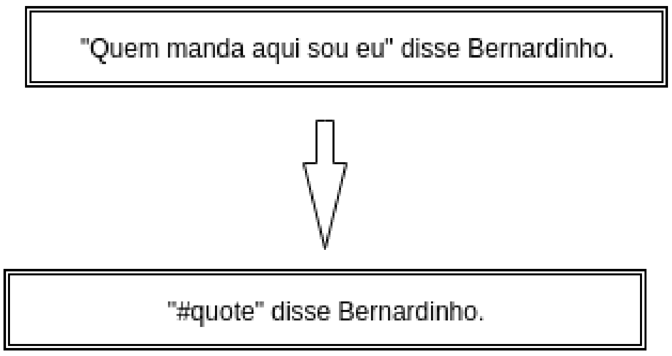

"Quem manda aqui sou eu" disse Bernardinho.
"#quote" disse Bernardinho.

Figure 4.6: Token contraction example.

The idea behind contracting tokens is to minimize distances between related tokens in a bigger sequence and the created new token defines a common representation to the contraction that is shared by all contractions performed in the input text. Performing those tokens contractions also enables our window classification approach to be performed, because since we are dealing with sequences of variable lengths, it brings us a strong way to make our input examples have a fixed size.

Although token contraction of variable length sequences is useful to make our input sequences have a fixed length, there are still two cases in which the examples may not respect our fixed size approach.

1. When the token being considered is near the sentence beginning and the number of tokens before it is smaller than the window length.

2. When the token being considered is near the sentence ending and the number of tokens after it is smaller than the window length.

To figure out those cases, we add special tokens called "*#pad*" to the window, to the input example. Padding tokens are added to the position where the real tokens should be found, and the number of padding tokens added is the number of missing tokens. In the example of Figure 4.7, the window of the "*#quote*" token is encoded with a window of size 5 ($c = 5$ tokens for the left side and $c = 5$ tokens for the right side). It exemplifies the cases when a "*#pad*" token is added to encode a token window.

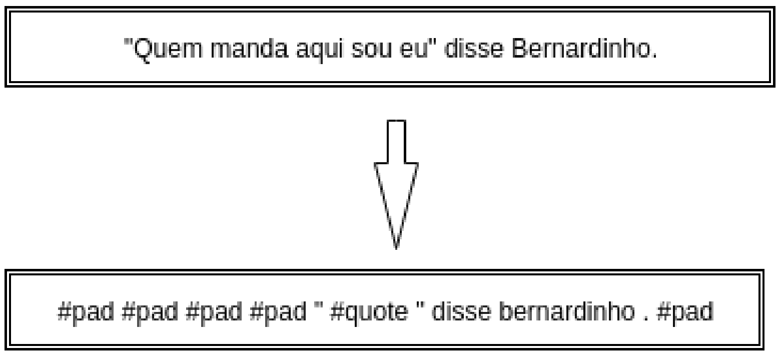

"Quem manda aqui sou eu" disse Bernardinho.
#pad #pad #pad #pad " #quote " disse bernardinho . #pad

Figure 4.7: Padding tokens example.

4.4.2 Extending The Window Definition

In most of our models, applying the definition of token windows above is sufficient to achieve our goals. However, in the *Quotation Attribution* subtask it is necessary to extend our token window definition to a more generalized form of window of tokens windows. More precisely, in this setting, a token is classified by the window of token windows that surrounds it.

In this setting, the size of the window is counted by the number of mentions seen from the quotation and going in both directions (left and right directions). Unlike the previous definition, the window of tokens is not necessarily symmetric, that is the size of the window can be different to the left and right directions.

The idea behind this encoding is to add to the input x_i of our model, the context of all mentions that must be considered when deciding which of them is the author of the quote being considered.

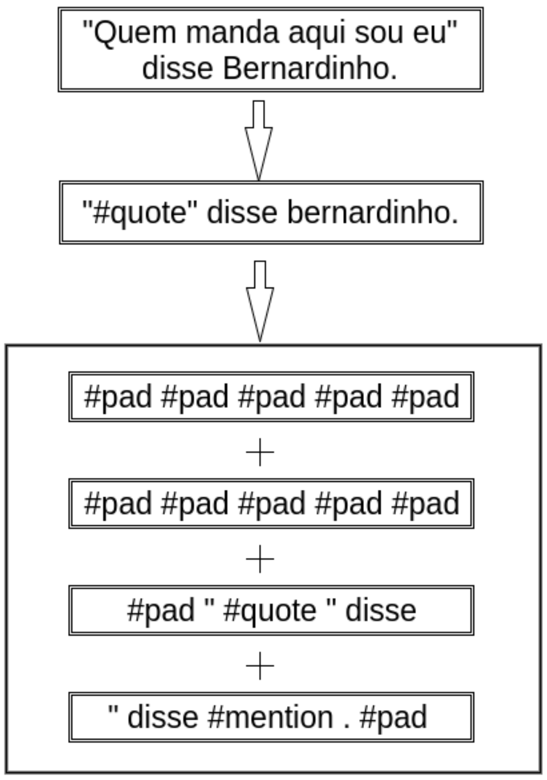


Figure 4.8: Window of token windows.

Figure 4.8 shows an illustrative example of an input encoded by this setting. In the example, , we used a window of 2 mentions to the left direction and 1 token mention to the right direction of the quotation. The window size of each token in this composition is 2 tokens to each side.

4.4.3 Illustrative Examples

To exemplify our approach, we present one illustrative example for each subtask in the Quotation Extraction and Attribution task decomposition.

The first step to solving our problem is the task of detecting quotations from the raw text. The machine learning model receives as input a window of tokens that surrounds the quotation candidates found in the input text and outputs a Boolean flag indicating if a given candidate is, actually, a quote or not. Figure 4.9 illustrates the input and output for the Quote Identification subtask.

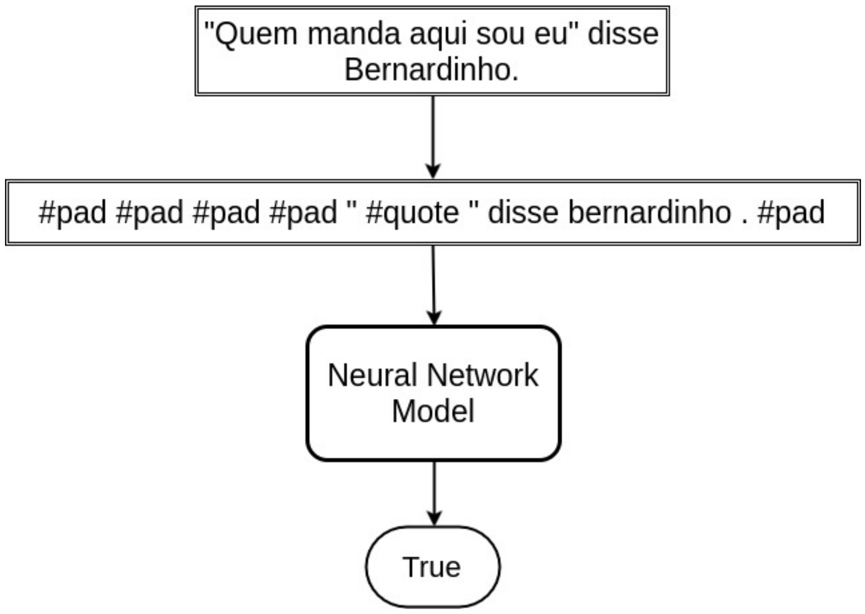

"Quem manda aqui sou eu" disse Bernardinho.
#pad #pad #pad #pad " #quote " disse bernardinho . #pad
Neural Network Model
True

Figure 4.9: Quote Identification subtask example.

The second step is the task of detecting author candidates from the raw text. In this step, the machine learning model receives as input a window of tokens that surrounds the mentions heads candidates found in the text and outputs a Boolean flag indicating if a given candidate is, actually, a mention or not. Figure 4.10 illustrates the input and output for the Author Candidates Identification subtask.

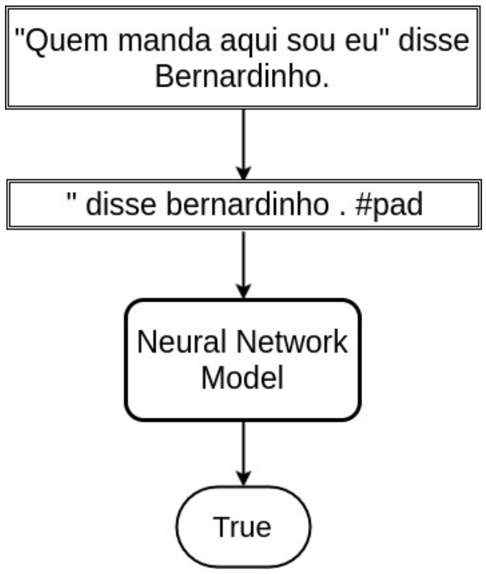

"Quem manda aqui sou eu" disse Bernardinho.
" disse bernardinho . #pad
Neural Network Model
True

Figure 4.10: Author Candidates Identification subtask example.

Finally, the last step is the task of linking authors to their quotations. This task is modeled as a multiclass classification task. Our Machine Learning model receives as input a set of tokens windows, which are composed of a quote window and its authors candidates windows. The output of our model uniquely identifies the position of the author. This position is given in an already explained relative distance (section 3.4), in which the distance is counted in numbers of mentions seen between a quotation and its author. As more than 98% of our quotations examples in the training set, belong to the classes "-1", "-2", "-3" and "$+1$", in this work we limit ourselves to dealing with this 4 classes, since these class imbalance problem would force our classifier to be biased towards the 4 majority classes (23, 5, 48, 27).

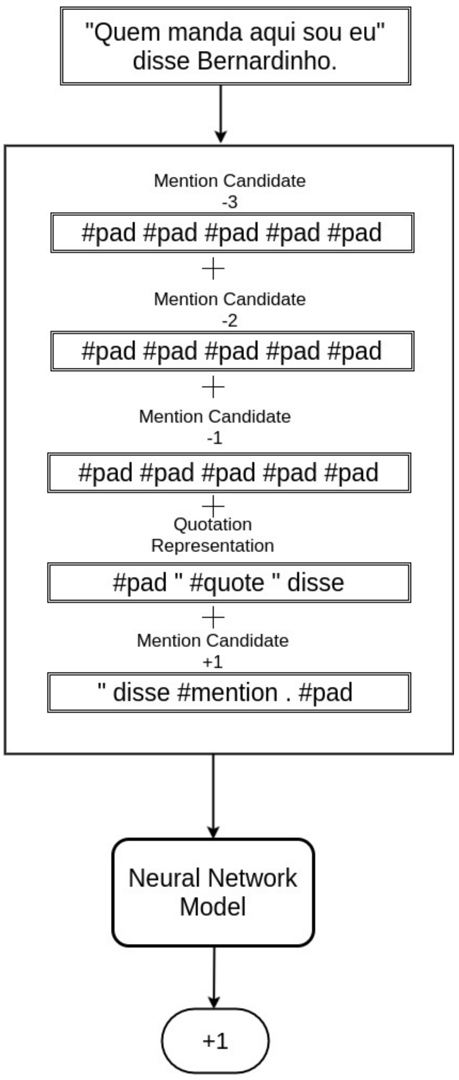


Figure 4.11: Quotation Attribution subtask example.

Figure 4.11 illustrates the input and output for the Quotation Attribution subtask. In the example, the input is encoded as it is described in Section 4.4.2.

4.5

Selecting Candidates to Quotations and Authors

Given a news document, our goal is to extract quotes and author candidates from the raw text and then link those extracted quotations to their respective authors. Our methodology, to address this objective, is to use machine learning models to detect, from a set of mentions and quotes candidates, the set of candidates that are truly mentions and quotes. Then this set of identified mentions and quotes is linked in an additional model. Therefore, a procedure to find quotations and mentions candidates is thus necessary to perform our methodology.

4.5.1
Quotation Candidates Extraction

To extract quotations candidates to feed our machine learning model, we use a rule based procedure that is able to retrieve from a given document a set of quotations candidates. Our methodology is based on the rule-based approach presented by Fernandes, in 2012, (15), and can capture 99.8% of the quotations found in a given document. It is important to notice that our ruled-based approach raises lots of false positive, adding to the set of quotation candidates a mean of almost three wrong examples for one quote example.

Our ruled extractor works in three steps looking for patterns. In the first step, it looks for a pattern that indicates the beginning of a quote. The second phase, it looks for patterns that indicate the end of a quotation. Finally, in the last step, the extractor uses the bounds, found in the previous steps, to extract the candidates.

4.5.1.1
Quote Beginning Rules

To find the start of a quotation, our extractor assigns a tag ’S’ to any token that is compatible with one of the two rules:

1. An ’S’ mark is assigned to a token that appears after any quotation marks (’ " -), except when a number precedes the quotation mark. The restriction handles with lists that can appear in a document. Figure 4.12 shows an illustrative example of this first rule. In the example, each "S" mark is subscripted after the token.

- **É**$_S$ a primeira vez no planeta que um time brasileiro paga sozinho quase 10 milhões de euros e não tem o atleta - **lamenta**$_S$ o mandatário alvinegro.

Figure 4.12: Quote beginning first rule example.

2. An 'S' mark is assigned to tokens after a period or question marks (.,?) followed by a proper noun and a colon (:). If a token follows the pattern but is a proper noun, then the "S" mark is not assigned. This last restriction deals with cases such as soccer team squad lists. Figure 4.13 shows an illustrative example of this second rule. In the example, each "S" mark is subscripted after the token.

(...) Copa do Mundo de 1958. GLOBOESPORTE.COM: O$_S$ Garrincha foi reprovado no exame psicológico, mas acabou sendo um de os destaques da Copa de 1958.

Figure 4.13: Quote beginning second rule example.

Figure 4.14 shows an illustrative example of these rule exceptions in a result position list of an Olympic game.

1 - Estado Unidos: 6m05s34
2 - Holanda: 6m07s22
3 - Romênia: 6m07s36

Figure 4.14: First rule exception example.

4.5.1.2
Quote Ending rules

To find the quotation end mark, our extractor uses marks taken in the first step and looks for the following patterns:

1. Starting from a quote token ('"-) followed by an "S" mark, the extractor reads all tokens until it reaches another quote token ('"-) or until it finds the end of the given document. Then an 'E' mark is assigned to the token that appears before the quote token or before the end of the document. If the quotation token being considered is a dash (-), the last dash can not be preceded by a token "ex" or followed by a personal pronoun. Those restrictions deal with cases where dashes are not used

as quotations delimiters, e.g. prefixes (*ex-jogador*) and enclisis (*procurá-lo*). Figure 4.15 presents an illustrative application of this rule. In the example, "S" and "E" marks are subscripted.

Roth elogia Roger e diz que armador tem que '**ir**$_S$ para o **sacrifício**$_E$'.

Figure 4.15: Quote end first rule example.

2. Starting from a proper name followed by a colon (":") and a token marked as a quotation beginning, the extractor reads all tokens until it reaches a sequence of tokens in the following order:

   I. a period or a question token ("." or "?")

   II. a proper name

   III. a colon(":")

   If this sequence of tokens is not found, the extractor reads all tokens until it reaches the end of the given document. Then an "E" mark is assigned to the period or question mark tokens or the token before the end of the document. Figure 4.16 presents an illustrative application of this rule. In the example, "S" and "E" marks are subscripted.

GLOBOESPORTE.COM: **Qual**$_S$ a importância da conquista da Copa do Mundo de 1958**?**$_E$ João Havelange: (...)

Figure 4.16: Quote end second rule example.

4.5.2 Mention Candidates Extraction

In the same way that our Quotation Extractor needs to identify quotes candidates to be filtered as correct or incorrect examples, our Author Candidates Identification layer also needs to extract mentions candidates to serve as input to our machine learning model.

Following the works of Moschitti, 2013 (44) and Peng, 2015 (38) we are concerned in finding the heads tokens of the mentions. This goal is achieved by using the constituency parse-tree, NER and POS tags of each given document to identify the mentions heads candidates. Thus, given a document and its tags, we consider as a mention head candidate any token that is marked as:

1. a beginning of a noun phrase (B-NP) on the constituency parse-tree.
2. any first token tagged as a Person or an Organization in NER tags.
3. any token tagged as a personal pronoun in POS tags.

So far, this approach would recover a great amount of mentions heads candidates that, in their majority, lie too far from quotations, in a given text. Since our models only deal with authors within a maximum distance of 4 mentions from a given quote, and we are only concerned with those mentions, we define a new distance metric measure. For this new metric, we determine a maximum range, for each side of a given quotation, in which a candidate can be extracted. This new metric is measured by counting, starting from a given quote and walking to the left or right side of this quote, the number of tokens tagged as a B-NP, a Person, an Organization or as a pronoun.

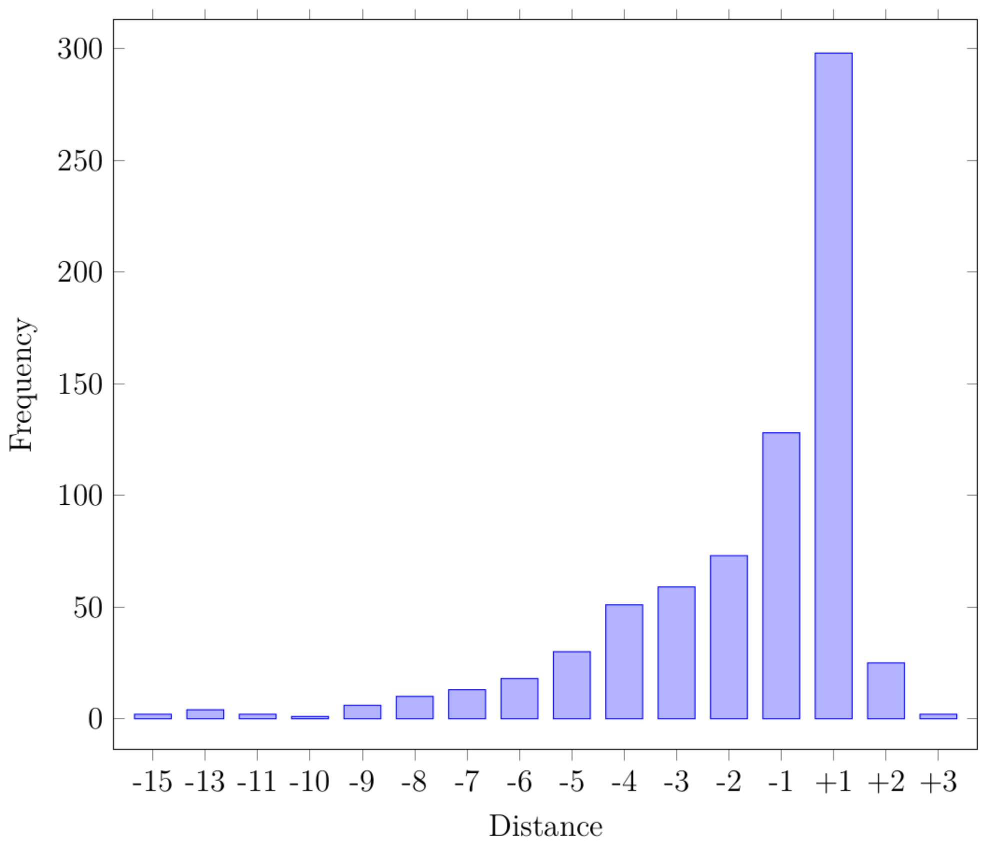


Figure 4.17: Distribution of mention reference types.

Figure 4.17 shows the distribution of this metric across the set of mentions that our model deals with. In this work, the maximum distance is defined as "-15" to the left side and "+3" to the right side of a given quotation.

4.6 Specific Layers Details

As discussed in section 4.1, our models can be divided into two kinds of layers:

1. Shared layers that are shared across all subtasks of the Quotation Extraction and Attribution task.

2. Subtask specific layers that are not shared and are defined to each subtask of the Quotation Extraction and Attribution task.

Since the representations learned by subtask specific layers are not shared between subtasks, it is possible to define different architectures in these layers for each subtask of our main problem.

Quote Identification and Quotation Attribution subtasks share the same architecture definition in their specific layers (the architecture definition is the same, but their parameters and outputs are not shared between the subtasks). In Figure 4.18 we represent the specific layers for those subtasks. The two first layers are fully connected hidden layers with Hyperbolic Tangent as their activation functions. The last layer is a standard Softmax layer responsible for outputting one score for each possible output.

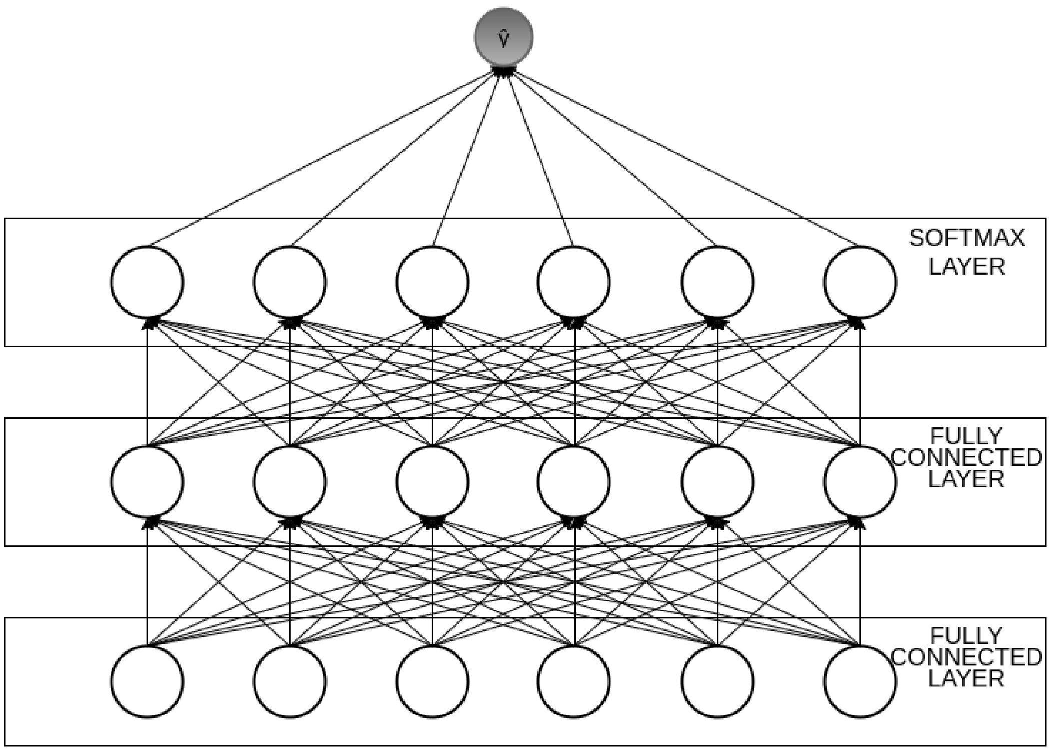


Figure 4.18: Specific layers representation.

Author Candidates Identification Specific Layers

Author Candidates Identification specific layers differ from other subtasks since three additional layers are added to the architecture:

1. A token extension representation layer.
2. An additional Bi-directional LSTM layer.
3. A Max layer

As discussed in section 4.2.2, it is possible to extend tokens representations, which are fed to our models, with additional discrete features. In the Author Candidates Identification subtask, tokens representations received from the shared part of our architecture are extended with three new features:

1. POS tags.
2. NER tags.
3. Common nominal mention heads bigram.

We retrieve the POS and NER tags from corpus annotations, and the feature of common nominal mentions heads bigram is retrieved from the dataset statistics of nominal mention heads. This feature is built by counting the number of times that a bigram appears, in the training set, being a mention head. Then, we build a list of bigrams in which their counts are at least two occurrences. Finally, every time we reach a sequence of two tokens that are in the bigrams list, these two tokens are tagged with "1", indicating that this sequence of two tokens is commonly used as nominal mentions heads. Otherwise, bigrams that are not on the list are tagged with "0". Figure 4.19 shows the nominal mention heads bigrams distribution from a set of 465 bigrams that appear with a distance of at most "-3" or "+1" mentions from any quotation in the corpus training set. As it is shown, in section 3.5, nominal mentions are only 20.91% of the total of mentions. From the set of nominal mentions, almost 80% of the nominal mentions bigrams appears at least three times as being a mention head and less than 11% of the nominal mention heads bigrams appear less than two times.

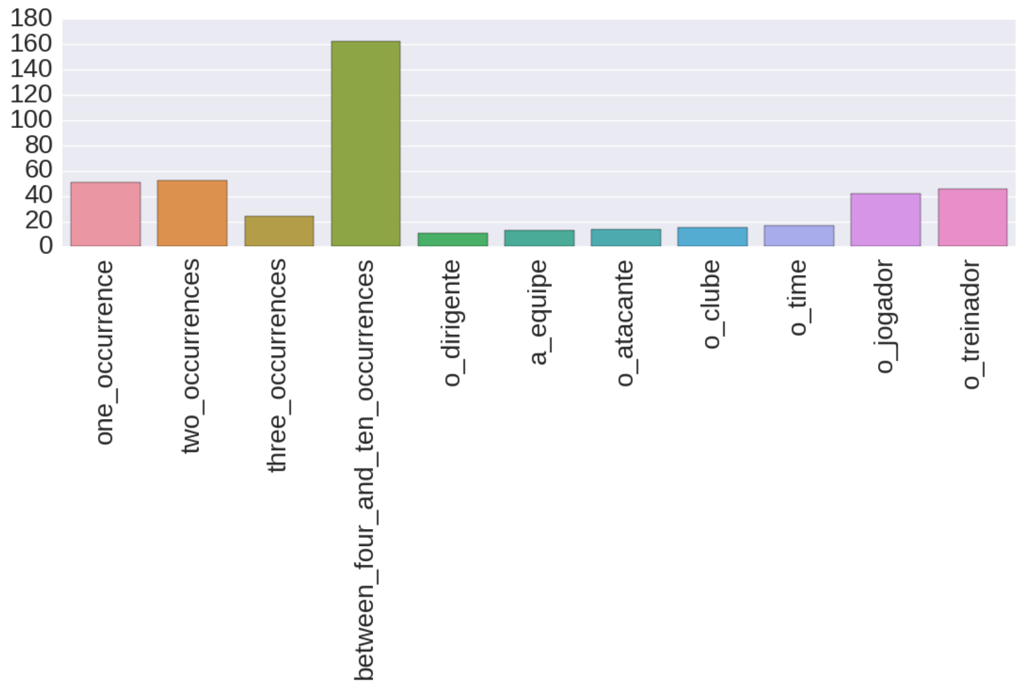


Figure 4.19: Nominal mentions heads bigram distribution.

After extending the tokens representations, these extended representations are fed to a second Bi-Directional LSTM layer. The objective of this layer is to capture how this additional information affects the sequential information of each token in the sequence. The outputs of the LSTM layer pass through a Max-layer to get a fixed size array representation of the sequence of tokens. Then, the vector representation passes through a fully connected hidden layer with Hyperbolic Tangent as its activation functions. Finally, the last layer is a standard SoftMax layer. Figure 4.20 illustrates the Author Candidates Identification subtask model specific layers.

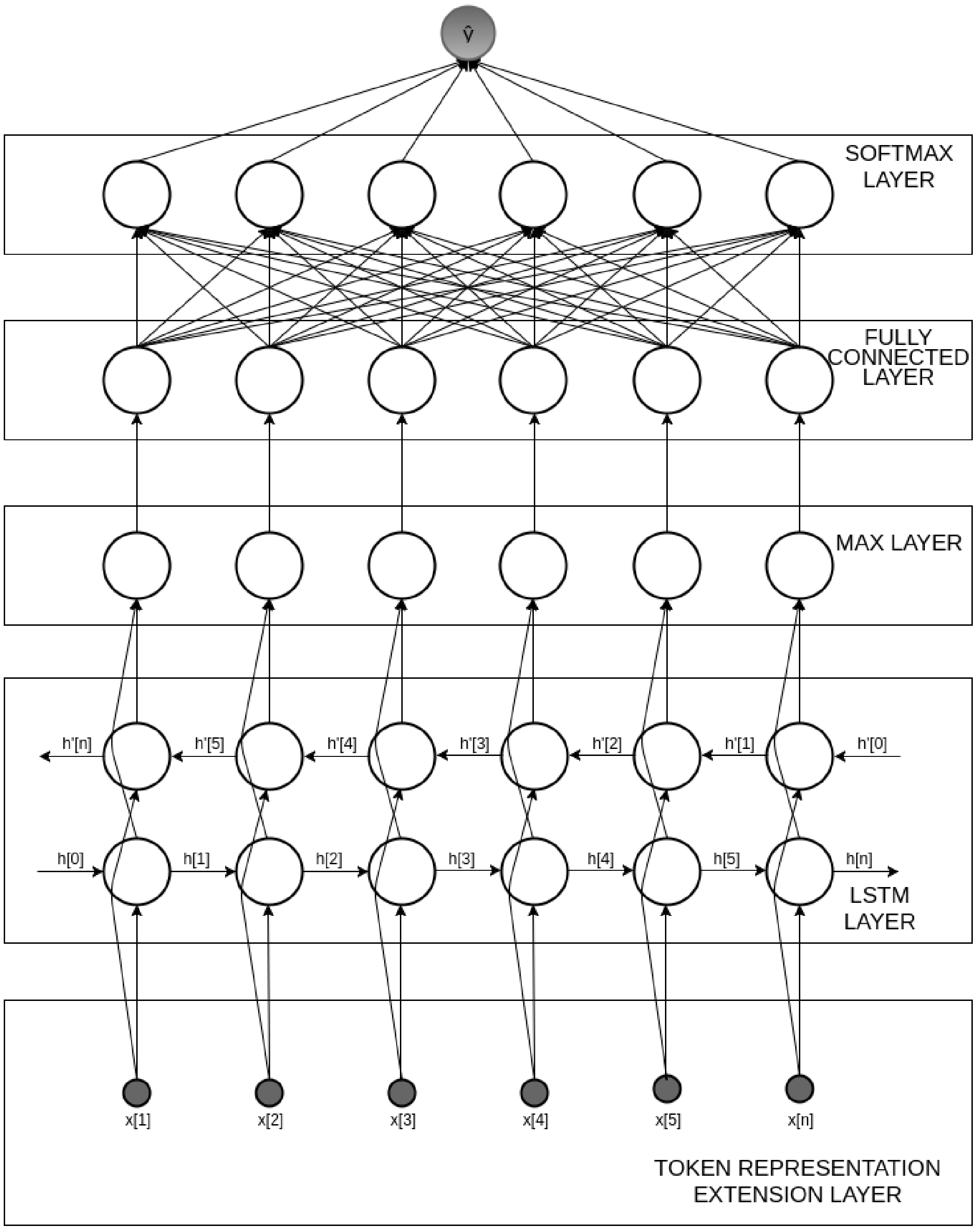


Figure 4.20: Author Candidates Identification specific layers.

4.7 Training

Our objective is to build Neural Network models that use training examples (x, y) generated from the corpus training set and its golden annotations to learn functions that fit the data. To achieve our goal, we use stochastic gradient descent (SGD) optimization algorithm with mini-batches of examples (1). Additionally, we use a per dimension adaptive learning rate technique, called

AdaDelta (46), to minimize the cross-entropy loss, added to a l_2 regularization term:

$$L(\theta) = (-\sum_x p(x) log[h(x)]) + \frac{\lambda}{2} \|\theta\|^2 \tag{4-16}$$

In equation 4-16, θ is the set of all parameters, h(x) is the output of the SoftMax layer, p(x) is the distribution where all probability mass is on the correct class and λ is a threshold defined to the l_2 normalization.

The models' parameters are initialized with pre-trained word embeddings, in the embedding layer, and the rest of the layers' parameters are initialized with random numbers. Training loss derivatives are back propagated to all parameters θ of the network (26), including the word embeddings.

Additionally, the models are trained with early stopping regularization.

To explore the parallelism of the operations, we train the network on a Graphics Processing Unit (GPU). A Python implementation using the Theano framework processes multiple inputs per time.

Multitasking learning

Multitask learning is the procedure of learning several tasks at the same time with the aim of mutual benefit (3, 7). Instead of learning one task at a time and breaking the problem into independent subproblems, in this work we joint train our subtasks models using the procedures by Collobert, in 2011 (8).

Our NNs automatically learn features for the subtasks in the deep layers of our architecture. It is thus reasonable to expect that when training our NNs, sharing deep layers would improve features produced by these deep layers, and thus improve generalization performance.

Training is achieved in a stochastic manner by looping over the tasks:

1. Select the next task.
2. Select a random training example for this task.
3. Update the Neural Network for this task by taking a gradient step with respect to this example.
4. Go back to 1.

5
Experiments

The task of Quotation Extraction and Attribution, as already discussed in Chapter 2, decomposes into three different subtasks:

1. Quote Identification.

2. Author candidates identification.

3. Quote Attribution.

This chapter presents the performance results for the Quotation Extraction and Attribution task and all its subtasks. We also present the experimental hyperparameters setup for the obtained results. First, in Section 5.1, we describe the corpus data split and how we use these splits in our work. In section 5.2, we define the evaluation metrics used in this work. Then, in Section 5.3, the model of each subtask is evaluated, and the corresponding parameter setting is presented. Finally, in Section 5.4, the whole system's performance is evaluated, compared and analyzed. We finish this Chapter with a brief analysis of our models' errors and results.

5.1
Corpus Split

We conduct our experiments on the GloboQuotes corpus. The corpus is divided into two groups:

– Training set

– Testing set

The size of each corpus set is presented in table 5.1.

Part	*Document*	*Sentences*	*Tokens*
Training	552	7.963	174.415
Testing	133	1.834	41.613

Table 5.1: Corpus set sizes.

Our models are trained and calibrated in the training set. In this work, we have also divided the training set into 5-folds, each fold containing about

20% of the training set. Then, the adjustment of the models' hyperparameters is performed running a grid search over the hyperparameters. The arrangement of the model's hyperparameters is evaluated by training the model on 80% (4-folds) of the training set, and the evaluation of the predictions is performed on 20% (1-fold) of the training set. The training and the evaluation are performed five times, leaving one different fold out of the training per time, and then the final scores are recovered by calculating the mean of the five scores obtained. This split-evaluation technique is commonly named as 5-fold cross-validation or leave-one-out cross-validation with five folds (25).

5.2 Evaluation Metrics

To evaluate the Quotation Extraction and Attribution task, the most common metrics are the *precision*, *recall* and F_1 metrics. In this work, we also use the accuracy metric, since the Quote Attribution task is modeled as a multiclass classification problem.

The precision metric measures the ability of a model to not label as positive a sample that is negative. More precisely, the precision metric is evaluated as

$$Precision = \frac{\#TruePositive}{\#TruePositive + \#FalsePositive} \tag{5-1}$$

, where $\#TruePositive$ is the number of examples whose correct labels are $True$ and that our model has classified as $True$ as well. In its turn, $\#FalsePositive$ is the number of examples classified as $True$ and whose correct label is $False$. In the case of Quote Identification subtask, this metric measures our model's ability to select quote candidates that are, indeed, quotes. In the Author Candidates Identification subtask, the precision metric measures our model's ability to select mention candidates that are, in fact, mentions. When evaluating the whole system, this metric evaluates the ability of our system to extract, from a given document, correct pairs of quotations and authors.

The recall metric measures the ability of a model to find all the positive samples that are present in data. Mathematically speaking, recall is measured as

$$Recall = \frac{\#TruePositive}{\#TruePositive + \#FalseNegative} \tag{5-2}$$

, where $\#TruePositive$ is the number of examples whose correct label is $True$ and that our model has classified as $True$ as well. In its turn, $\#FalseNegative$ is the number of examples classified as $False$, and whose correct label is $True$. In the case of Quote Identification task, the recall metric measures our model's ability of recover, from the set of quote candidates, the set of quotes examples. In Author Candidates Identification subtask, the recall measures our model's ability of recover, from the set of mention candidates, the set of mentions examples. When evaluating the whole system, the recall metric evaluates the ability of our system to extract, from a given document, pairs of quotes and authors that are present in the document.

The F_1 metric is a weighted average of the precision (described in equation 5-1) and recall (described in equation 5-3) metrics. This metric is evaluated as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{5-3}$$

The Accuracy is defined as the number of items categorized correctly divided by the total number of items. It simply measures what fraction of the examples results in correct classifications by a model. More formally, it is measured by:

$$Accuracy = \frac{\#CorrectClassifications}{\#TotalNumberOfExamples} \tag{5-4}$$

In this work, we use the Accuracy metric to evaluate the Quotation Attribution subtask.

5.3 Subtask Evaluations

In this section, we present the individual evaluations of our models, assessed in the test set. When it is possible, the individual evaluations are compared with the previous work on the same dataset (15).

In all subtasks, we use word vectors representation of size 50, and the output size of the shared LSTM layer equals to 400.

5.3.1 Quote Identification Evaluation

In the validation step, our model achieves an F_1 score of 94.16% for the hyperparameter settings that are presented in Table 5.2. The validation scores are presented in Table 5.3.

The quality of our model in the Quote Identification subtask, assessed in the test set, is shown in Table 5.4. Our model obtains an F_1 score of $95,04\%$, which is an error reduction of 59.74% compared to the ETL model, proposed by Fernandes in 2012 (15).

Hyperparameter	*Setting*
Learning Rate	0.1
Mini Batch Size	100
L_2 Normalization (λ Parameter)	0.01
Window Size	5
Hidden Layers Size	100
AdaDelta (ρ Parameter)	0.98
AdaDelta (ϵ Parameter)	1×10^{-6}

Table 5.2: Quote Identification Hyperparameter settings.

Model	*Precision%*	*Recall%*	F_1*%*
This Work	94.16	96.26	95.20

Table 5.3: Quote Identification scores assessed in the validation step.

Model	*Precision%*	*Recall%*	F_1*%*
This Work	**95.88**	**94,21**	**95,04**
ETL	85.25	90.24	87.68
Rule-Based baseline	30.18	99.51	46.31

Table 5.4: Performance of the Quote Identification subtask.

5.3.2 Author Candidates Identification Evaluation

In the validation step, our model achieves an F_1 score of 99.13% for the hyperparameter settings that are presented in Table 5.5. The validation scores

Hyperparameter	*Setting*
Learning Rate	0.1
Mini Batch Size	200
L_2 Normalization (λ Parameter)	0.02
Window Size	4
Hidden Layers Size	100
AdaDelta (ρ Parameter)	0.95
AdaDelta (ϵ Parameter)	1×10^{-6}

Table 5.5: Author Candidates Identification Hyperparameter settings.

are presented in Table 5.6.

The quality of our model in the Author Candidates Identification subtask, assessed in the test set, is presented in Table 5.7. The performance of our model, for this subtask, cannot be directly compared to the previous work, since there are no previous results on this subtask.

Since the main goal of Author Candidates Identification subtask is selecting author candidates for a given quotation, it is worth measuring the percentage of *authors* (mentions that are, indeed, the author of a quote) that our model can retrieve. This percentage is calculated by counting the total

Model	*Precision%*	*Recall%*	*F_1%*
This Work	99.13	99.13	99.13

Table 5.6: Author Candidates Identification scores assessed in the validation step.

Model	*Precision%*	*Recall%*	*F_1%*
This Work	**83.47**	**98.96**	**90.56**

Table 5.7: Performance of the Author Candidates Identification subtask.

number of authors presented in the output of our model, and dividing it by the total number of golden authors present in the input data. Table 5.8 presents the resulting measure for both the cross-validation step and for the corpus test set.

Partition	*Authors Recall%*
Validation Set	100.00
Test Set	97.32

Table 5.8: Percentage of authors retrieved from the corpus.

5.3.3 Quote Attribution Evaluation

The individual Quotation Attribution subtask performance is measured with the Accuracy metric. By now, the accuracy metric presented in this section is calculated only across the four classes (most frequent relative distances) that our system handles with. In the validation step, our model achieves an Accuracy score of 92.42% for the hyperparameter settings that are presented in Table 5.9.

Hyperparameter	*Setting*
Learning Rate	0.1
Mini Batch Size	100
L_2 Normalization (λ Parameter)	0.03
Quote Window Size	3
Mention Window Size	5
Hidden Layers Size	100
AdaDelta (ρ Parameter)	0.95
AdaDelta (ϵ Parameter)	1×10^{-6}

Table 5.9: Quote Attribution Hyperparameter settings.

The quality of our model in the Quote Attribution subtask, assessed in both validation and test set, is presented in Table 5.10.

Partition	*Accuracy%*
Validation	92.42
Test	94.6

Table 5.10: Performance of the Quote Attribution subtask.

5.4 Whole System Evaluation

The whole system performance is measured with the precision, recall, and F_1 metrics. For this analysis, we consider as true positive, pairs of quotations and authors that are correctly recovered from the corpus. We also consider as false positive, pairs of quotes and authors that are incorrectly recovered from the corpus. A false negative is a pair of quotation and author that our system can not recover from the corpus.

Model	*Precision%*	*Recall%*	F_1%
This Work	**90.70**	**88.20**	**89.43**
SP_WIS	83.24	71.49	76.80
ETL	69.44	73.17	71.26
Rule Based	64.35	67.8	66.03

Table 5.11: Performance of the whole system compared to the previous work.

Table 5.11 presents the performance, assessed in the test set, of the whole system compared to the previous work. Our whole system achieved an F_1 score of 89.43%, which is an error reduction of 54.43% compared to the SP-WIS model, proposed by Fernandes in 2012 (15).

5.5 Analysis

We observe that the division into less complex subtasks helps to solve the Quotation Extraction and Attribution problem.

Overall, our method outperforms the baseline system (15) in each subtask, and in the whole task. Also, our system solves the Author Candidates Identification subtask, which the previous system assumes as an already solved problem.

In order to gain insights about our models and about where we are propagating errors, we make a brief analysis of the residual errors for the Author Candidates Identification subtask and for the Quote Attribution subtask. Since, the missclassification rate, in the Quote Identification subtask is about 4% we have not identified any pattern for this subtask.

In the subtask of Quote Attribution, we could observe little confusion between the four classes, being the most common mistakes around the class

"-2" and its neighbors. This confusion between classes is represented in Table 5.12. It is possible to observe that most of the confusion is on the frontiers of the categories, and we can observe only one occurrence of confusion that occurred above the frontier (between classes "-3" and "-1"). When we analyze the window of tokens of the true author candidates, it is noticed that in 7 out of the 10 mistakes the "#UNK" token (defined in section 4.2.1) is present on the window.

Tags	**-3**	**-2**	**-1**	**+1**
-3	7	2	0	0
-2	1	15	3	0
-1	1	2	58	0
+1	0	0	1	92

Table 5.12: Confusion Matrix of the Quote Attribution subtask.

In the Author Candidates Identification subtask, most of our model's errors occur near to some "#UNK" token. This token appears in 31 out of 80 examples that our model mistakes. As expected, sequences of two tokens (e.g. "o jogador") that are tagged using the tagging scheme explained in section 4.6 are biased to be classified as mentions. Apparently, this tag introduced 35 miss-classifications in the test set. We present the confusion matrix on table 5.13.

Tags	**False**	**True**
False	1492	76
True	4	384

Table 5.13: Confusion Matrix of the Author Candidates Identification subtask.

These results suggest that that in order to enhance the performance of our model, it is necessary to find a better representation to "#UNK" tokens. They also suggest that, despite the fact that using statistics about mention bigrams helps to learn nominal mentions, it also introduces errors in classifications, biasing our model toward the tagged bigrams.

6
Conclusions

In this work, we present a novel Quotation Extraction and Attribution system using a unified neural network architecture. The experimental evaluation shows that our system outperforms the previous work (15) in each subtask, and in the whole task. The primary objective of our work is to learn a model that can assign, with high performance, an author for each quotation found in a given news document. In the end, the objective is met with a reasonable best F_1 score of $89,43\%$ for the test data. This score is achieved by finding a common dense vector representation to each token and modeling their interactions through the shared and specific layers of our neural networks. Our model only relies on dense features and is able to learn most of the features needed to perform predictions.

We believe that this work contributes towards a full Quotation Extraction and Attribution system. Nevertheless, we are aware that our methodology can be criticized. The main limitations can be summarized as follows.

– Over the years, the NLP community has developed considerable expertise in engineering useful NLP features for the tasks performed in this work. These features could help to improve our system performance.

– Although we do not design features by hand, our system relies on a significant amount of preprocessing to build the inputs of each system subtask. These preprocessing steps are based on rules from the task domain knowledge, which needs to be reviewed when considering to apply our methodology to new languages and domains.

– All subtasks of our system receive as input documents labeled with tags received or from the corpus either from the previous subtasks (e.g. POS tags are used to preprocess the data of the Quote Identification subtask), which can be considered as features cascading through tasks. Feature cascading is not desired because it can propagate error and noise from one task to another.

– Despite the ability of our system to learn features to solve the task, it relies on the most common mention heads tags as input to solve the Author Candidates Identification subtask. Those tags are, indeed, features designed based on the corpus statistics of mention heads.

  – Since our models do not deal with relative distances bigger than "-3", for the left side of a quote, and "+1" for the right side of a quote, we renounce to find the actual author of about $1,25\%$ of the dataset.

The usual methods to solve the Quotation Extraction and Attribution task consist of extracting a rich set of hand designed features from a given text document. In this work, instead, we avoid using these hand engineered features. For this purpose, we deliver unlabeled data prior knowledge to our models and train them to learn relevant representations to each token, which are shared across the whole system. Avoiding the use of hand designed features made our model easily adaptable to other languages and domains. Moreover, it helps to save a considerable amount of time to implement these features. Additionally, allowing the network to learn hidden representations by itself made our system unbiased towards prior knowledge. It is commonly reported that the learned model is highly dependent on the distances features and to the list of reported speech verbs (15, 41). However, even considering these advantages, we recognize the need to investigate the impact of extending our methodology with those features, and further work is needed in this direction.

Although we do not design features by hand, our model's inputs are still preprocessed. This preprocessing step is simple and easy to reproduce. In general, this step consists only in building windows of some specific tokens that are found in a given news document. The most complex preprocessing is made in the Quote Identification subtask, in which, for finding and contracting quote candidates, it relies on a set of rules. As explained in Chapter 4, this step of preprocessing is responsible for reducing the input dimensionality and focusing the models on the text regions that we are interested in. Therefore, our preprocessing step is kept simple and invariant to language and domain.

Despite the simplicity of the preprocessing procedures, they rely on external information (recovered from the corpus tags) or from the previous subtasks outputs. The use of those external inputs can propagate errors, and in a certain way, give to our models a prior knowledge. However, all external data, which we use to build the inputs of our models, are obtained from well known NLP tasks. Also, those tasks could be inserted in our stack, in an attempt of further improvement of our system. The cascading error propagation through each subtask is the price of our strategy of splitting the whole task into three simpler subtasks. Furthermore, we can observe in our results that these errors propagations are not a big problem since the learned individual models are very accurate.

In the Author Candidate Identification subtask, our model also relies on an engineered feature as input. This engineered feature is based on the counts of mention heads bigrams, presented in the corpus training set, and is straightforward and easy to adapt and reproduce to other datasets and languages. Furthermore, our system uses this engineered feature on a subtask that the related works (15, 41) commonly bypass (when this kind of information is used in the methodology) and uses the golden corpus annotations to solve the problem. Moreover, it is a common practice in the deep learning literature to serve this kind of feature to the model, using a list of well-known mentions, commonly named as gazetteers. In general, this list is built using external sources of information, which make it more complicated and difficult to reproduce (8, 7, 32, 38, 2, 24).

The last step in our task decomposition is the Quote Attribution subtask. In this work, this subtask is modeled as a multi-class classification problem, in which each possible label for a given quote is the author relative distance from the quote. As we noticed, almost $98,22\%$ of the quotations presented in the dataset belong to the classes "-3", "-2", "-1" and "+1". The remaining $1,78\%$ is distributed between 4 classes, each with too few numbers of examples. Since this class imbalance problem would force our classifier to be biased towards the four majority classes (23, 5, 48, 27), our system is limited to dealing with them, accepting the fact that our model will not be able to handle the other four rare classes.

Future Work

An interesting line of future work is to combine our neural network based classifier with structured models as an attempt to improve our performance. As mentioned before, we left as future work a study on the impact of using hand-designed features, such as the distance of each token to the nearest quotation, in our models. Also, there is still room for improvement in our methodology, such as finding better ways to dealing with "#UNK" tokens and adding the subtasks of POS and NER tagging to our system stack.

Bibliography

[1] BENGIO, Y. CoRR. **Practical recommendations for gradient-based training of deep architectures**, journal, v.abs/1206.5533, 2012.

[2] BONADIMAN, D.; SEVERYN, A. ; MOSCHITTI, R. **Deep Neural Networks for Named Entity Recognition in Italian**, 2015.

[3] CARUANA, R. Machine Learning. **Multitask learning**, journal, v.28, n.1, p. 41–75, 1997.

[4] CHEN, D.; MANNING, C. **A fast and accurate dependency parser using neural networks.** In: PROCEEDINGS OF THE 2014 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] CHUNG, Y.-A.; LIN, H.-T. ; YANG, S.-W. **Cost-aware pre-training for multiclass cost-sensitive deep learning.** In: Kambhampati, S., editor, IJCAI, p. 1411–1417. IJCAI/AAAI Press, 2016.

[6] COLLINS, M. **Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms.** In: PROCEEDINGS OF THE ACL-02 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING-VOLUME 10, p. 1–8. Association for Computational Linguistics, 2002.

[7] COLLOBERT, R.; WESTON. **A unified architecture for natural language processing: deep neural networks with multitask learning.** In: ICML'08 PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 160–167. ACM, 2008.

[8] COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K. ; KUKSA, P. P. CoRR. **Natural language processing (almost) from scratch**, journal, v.abs/1103.0398, 2011.

[9] DE LA CLERGERIE, É.; SAGOT, B.; STERN, R.; DENIS, P.; RECOURCÉ, G. ; MIGNOT, V. **Extracting and visualizing quotations**

from news wires. In: LANGUAGE AND TECHNOLOGY CONFERENCE, p. 522–532. Springer, 2009.

[10] DE MORAIS, L. A. D. F.; NUNES, S. S. ; OTHERS. **Automatic extraction of quotes and topics from news feeds**. In: DSIE09-4TH DOCTORAL SYMPOSIUM ON INFORMATICS ENGINEERING, 2009.

[11] DOS REIS SILVA, R. **Direct and Indirect Quotation Extraction for Portuguese**. 2017. Master's thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[12] DOS SANTOS, C. N. **Entropy Guided Transformation Learning**. 2009. Doctoral thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[13] DYER, C.; BALLESTEROS, M.; LING, W.; MATTHEWS, A. ; SMITH, N. A. CoRR. **Transition-based dependency parsing with stack long short-term memory**, journal, v.abs/1505.08075, 2015.

[14] ELSON, D. K.; MCKEOWN, K. R. **Automatic attribution of quoted speech in literary narrative**. In: PROCEEDINGS OF THE TWENTY-FOURTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, AAAI'10, p. 1013–1019. AAAI Press, 2010.

[15] FERNANDES, W. P. D. **Quotation Extraction for Portuguese**. 2012. Master's thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[16] GRAVES, A. CoRR. **Generating sequences with recurrent neural networks**, journal, v.abs/1308.0850, 2013.

[17] GRAVES, A.; FERNÁNDEZ, S. ; SCHMIDHUBER, J. **Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition**, p. 799–804. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[18] GRAVES, A.; FERNÁNDEZ, S. ; SCHMIDHUBER, J. **Bidirectional lstm networks for improved phoneme classification and recognition**. In: PROCEEDINGS OF THE 15TH INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS: FORMAL MODELS AND THEIR APPLICATIONS - VOLUME PART II, ICANN'05, p. 799–804, Berlin, Heidelberg, 2005. Springer-Verlag.

[19] GRAVES, A.; MOHAMED, A. ; HINTON, G. E. CoRR. **Speech recognition with deep recurrent neural networks**, journal, v.abs/1303.5778, 2013.

[20] HINTON, G. E. Trends in Cognitive Sciences. **Learning multiple layers of representation**, journal, v.11, n.10, p. 428–434, 2007.

[21] HOCHREITER, S.; SCHMIDHUBER, J. Neural Comput. **Long short-term memory**, journal, v.9, n.8, p. 1735–1780, Nov. 1997.

[22] HUANG, E. H.; SOCHER, R.; MANNING, C. D. ; NG, A. Y. **Improving word representations via global context and multiple word prototypes**. In: PROCEEDINGS OF THE 50TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: LONG PAPERS - VOLUME 1, ACL '12, p. 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[23] KHAN, S. H.; BENNAMOUN, M.; SOHEL, F. A. ; TOGNERI, R. CoRR. **Cost sensitive learning of deep feature representations from imbalanced data**, journal, v.abs/1508.03422, 2015.

[24] KHASHABI, D. **On the recursive neural networks for relation extraction and entity recognition**. 2013.

[25] KOHAVI, R. **A study of cross-validation and bootstrap for accuracy estimation and model selection**. p. 1137–1143. Morgan Kaufmann, 1995.

[26] LECUN, Y.; BOTTOU, L.; ORR, G. B. ; MÜLLER, K. R. **Efficient BackProp**, p. 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[27] LING, C. X.; SHENG, V. S. **Cost-Sensitive Learning and the Class Imbalance Problem**, p. 231–235. Springer US, Boston, MA, 2010.

[28] MAMEDE, N.; CHALEIRA, P. **Character Identification in Children Stories**, p. 82–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[29] MIKOLOV, T.; CHEN, K.; CORRADO, G. ; DEAN, J. CoRR. **Efficient estimation of word representations in vector space**, journal, v.abs/1301.3781, 2013.

[30] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. ; DEAN, J. CoRR. **Distributed representations of words and phrases and their compositionality**, journal, v.abs/1310.4546, 2013.

[31] MIWA, M.; BANSAL, M. CoRR. **End-to-end relation extraction using lstms on sequences and tree structures**, journal, v.abs/1601.00770, 2016.

[32] NGUYEN, T. H.; SIL, A.; DINU, G. ; FLORIAN, R. CoRR. **Toward mention detection robustness with recurrent neural networks**, journal, v.abs/1602.07749, 2016.

[33] NOGUEIRA DOS SANTOS, C.; L.MILIDIÚ, R. ; P.RENTERIA, R. **Portuguese part-of-speech tagging using entropy guided transformation learning**. In: COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, p. 143–152. Springer, 2008.

[34] PARETI, S.; O'KEEFE, T.; CURRAN, J.; KOPRINSKA, I. ; HONNIBAL, M. **A sequence labelling approach to quote attribution**. In: PROCEEDINGS OF THE 2012 JOINT CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING AND COMPUTATIONAL NATURAL LANGUAGE LEARNING, p. 790–799. ACM, 2012.

[35] PARETI, S.; O'KEEFE, T.; KONSTAS, I.; CURRAN, J. R. ; KOPRINSKA, I. **Automatically detecting and attributing indirect quotations.** In: EMNLP, p. 989–999, 2013.

[36] PASCANU, R.; GÜLÇEHRE, Ç.; CHO, K. ; BENGIO, Y. CoRR. **How to construct deep recurrent neural networks**, journal, v.abs/1312.6026, 2013.

[37] PASCANU, R.; MIKOLOV, T. ; BENGIO, Y. CoRR. **Understanding the exploding gradient problem**, journal, v.abs/1211.5063, 2012.

[38] PENG, H.; CHANG, K. ; ROTH, D. **A joint framework for coreference resolution and mention head detection**. In: CONLL, p. 10, University of Illinois, Urbana-Champaign, Urbana, IL, 61801, 7 2015. ACL.

[39] PENNINGTON, J.; SOCHER, R. ; MANNING, C. D. **Glove: Global vectors for word representation**. In: EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 1532–1543, 2014.

[40] POULIQUEN, B.; STEINBERGER, R. ; BEST, C. **Automatic detection of quotations in multilingual news**. In: PROCEEDINGS OF RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING, p. 487–492, 2007.

[41] QUINTÃO, M. **Quotation Extraction for Portuguese Corpora**. 2014. Master's thesis - Department of Electrical and Conputer Egineering, Técnico Lisboa.

[42] RAMSHAW, L. A.; MARCUS, M. P. CoRR. **Text chunking using transformation-based learning**, journal, v.cmp-lg/9505040, 1995.

[43] TURNEY, P. D.; PANTEL, P. CoRR. **From frequency to meaning: Vector space models of semantics**, journal, v.abs/1003.1141, 2010.

[44] URYUPINA, O.; MOSCHITTI, A. **Multilingual mention detection for coreference resolution**. In: INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING, p. 100–108, 2013.

[45] WESTON, J.; RATLE, F. ; COLLOBERT, R. **Deep learning via semi-supervised embedding**. In: PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, ICML '08, p. 1168–1175, New York, NY, USA, 2008. ACM.

[46] ZEILER, M. D. CoRR. **ADADELTA: an adaptive learning rate method**, journal, v.abs/1212.5701, 2012.

[47] ZEYER, A.; DOETSCH, P.; VOIGTLAENDER, P.; SCHLÜTER, R. ; NEY, H. CoRR. **A comprehensive study of deep bidirectional LSTM rnns for acoustic modeling in speech recognition**, journal, v.abs/1606.06871, 2016.

[48] ZHOU, Z.-H.; LIU, X.-Y. Computational Intelligence. **On multi-class cost-sensitive learning.**, journal, v.26, n.3, p. 232–257, 2010.

[49] ZITOUNI, I.; FLORIAN, R. **Mention detection crossing the language barrier**. In: PROCEEDINGS OF THE CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, EMNLP '08, p. 600–609, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.