TÉCNICO LISBOA

Co-Reference Resolution in Portuguese and Spanish Texts

Nádia Sofia Mawjood Fernandes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Henrique Daniel de Avelar Lopes Cardoso

Examination Committee

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. Ricardo Daniel Santos Faro Marques Ribeiro

January 2021

Acknowledgments

I would like to thank my supervisors, Professor Bruno Martins and Professor Henrique Cardoso, for all their guidance, support, and insight which were fundamental throughout this thesis and made it possible to complete it.

I am grateful to my mother, who despite being no longer present, always supported and loved me in all my choices, made it possible for me to be where I am now and made me who I am, and for that I will always be grateful. I am also grateful to my father and brother, for all the support, confidence and strength they have given me to move on and overcome any difficulties. For always caring, wishing me well and having a hand ready to help me achieve my goals. Without them, this accomplishment would not be possible.

To Alexandra, Gonçalo and Ana, for staying with me through all these years and being my support in and out of the academic life, I really cherish your friendship and I am grateful for it. To Cecília, Mara, Daniel, José, Marco and João who have been with me for the past 5 years and with whom I share the best memories. To Manuel, Francisco and Tomás, who taught and helped me a lot over the years, and shared this stage with me always encouraging each other. To Joana, Mariana, Catarina, Duarte, Leandro, Artur and Bernardo whom I have had the luck to get to know better in the last years and have been present and untiring since then. To each person who passed through CPLEIC, who made this my home and with whom I share the best stories to remember in the future, a huge thank you.

I would also like to express my gratitude to Adriana, for her availability and advice during this work.

Finally, it is important to acknowledge the Foundation for Science and Technology (FCT), for supporting through the project grant with reference POCI/01/0145/FEDER/031460 (DARGMINTS).

To each and every one of you – Thank you.

Dedicated to my mother

Abstract

Co-reference resolution is a task focused on identifying the expressions in a text referring to the same entity. It has attracted a great deal of attention due to its importance in language understanding and as a subtask for other Natural Language Processing problems. The current state-of-the-art approaches are based on the supervised training of deep neural networks, which presents a challenge for less-resourced languages, such as Portuguese. In this work we propose a state-of-the-art neural co-reference resolution model for Portuguese and Spanish texts. The developed model explores a cross-lingual learning approach, aligning Portuguese and Spanish word embeddings in a single vector space, and training simultaneously with data from both languages, tackling the problem of Portuguese being a less-resourced language. Our model builds on a previous neural co-reference resolution system, developed and tuned for English data, which we adapt to the cross-lingual scenario. The proposed model shows that a cross-lingual learning approach with Portuguese and Spanish data achieves promising results, close to the ones obtained from the respective monolingual models.

Keywords

Co-Reference Resolution, Cross-Lingual Learning, Deep Learning, Natural Language Processing

Resumo

Resolução de co-referências é uma tarefa focada em identificar as expressões num texto referentes a uma mesma entidade. Tem atraído muita atenção devido à sua importância na compreensão da linguagem e como uma subtarefa para outros problemas de Processamento de Língua Natural. As abordagens atuais do estado da arte são baseadas no treino supervisionado de redes neuronais profundas, o que representa um desafio para línguas com poucos recursos, como o Português. Neste trabalho nós propomos um modelo neuronal para resolução de co-referências em textos Portugueses e Espanhóis, alinhado com o estado da arte. O modelo desenvolvido uma abordagem de aprendizagem multilíngue, alinhando os embeddings de Português e Espanhol num espaço vetorial comum, e treinando simultaneamente com dados de ambas as línguas, combatendo o problema do Português ser uma língua com menos recursos. O nosso modelo baseia-se num sistema neuronal para resolução de co-referências pré-existente, desenvolvido e ajustado para dados em Inglês, que nós adaptamos para o cenário multilíngue. O modelo proposto mostra que uma abordagem baseada em aprendizagem multilíngue com dados em Português e Espanhol atinge resultados promissores, próximos dos resultados obtidos pelos respectivos modelos monolíngue.

Palavras Chave

Resolução de Co-Referências, Aprendizagem Multilíngue, Aprendizagem Profunda, Processamento de Língua Natural

Contents

List of Figures

List of Tables

Acronyms

BERT	Bidirectional Encoder Representations from Transformers
BiLM	Bidirectional Language Model
BiLSTM	Bidirectional Long Short-Term Memory
BiRNN	Bidirectional Recurrent Neural Network
CBOW	Continuous Bag of Words
CSLS	Cross-Domain Similarity Local Scaling
CNN	Convolutional Neural Network
CORP	Co-Reference Resolution for Portuguese
ELMo	Embeddings from Language Models
FFNN	Feed-Forward Neural Network
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Masked Language Model
MLP	Multi-Layer Perceptron
MSE	Mean-Squared Error
mUSE	Multilingual Universal Sentence Encoder
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
NN	Neural Network
PCA	Principal Component Analysis
POS	Part-of-Speech
RNN	Recurrent Neural Network

SGD Stochastic Gradient Descent

SVD Singular Value Decomposition

1

Introduction

Contents

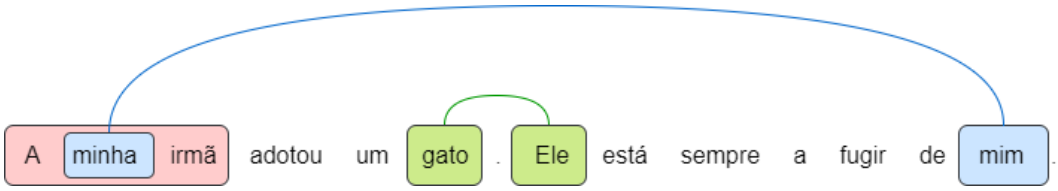

Figure 1.1: Co-reference resolution example for Portuguese.

Natural Language Processing (NLP) is a scientific field focused on giving computing machines the ability to process, interpret and generate natural language (i.e. human language). Hence, NLP involves several tasks, such as the co-reference resolution task.

Co-reference resolution consists in identifying the expressions in a text (e.g. pronouns and nouns) that refer to the same entity. Those referring expressions are called mentions and the entity to which they refer to it is called the referent. A group of mentions with the same referent is called a co-reference chain or a cluster. Figure 1.1 presents a co-reference resolution example. The mentions are identified by rectangles and the arcs express a co-reference relation. The different colours represent the clusters. Similarly to the example, the goal of a co-reference resolution system is to output all the co-reference clusters of a given text.

1.1 Motivation

The problem of co-reference resolution has been studied for many years. However, recently, researchers started testing cutting-edge deep learning techniques to solve it, re-gaining the interest in the subject. Co-reference resolution already has several existent solutions trained and tested over high-resourced languages, such as English, achieving great results and improving the state-of-the-art. However, few experiments with neural models have been done for less-resourced languages such as Portuguese, due to the smaller amount of data to train the models.

Cross-lingual learning is a promising solution to tackle the problem of less-resourced languages, based on using data from other languages to train the model. To the best of our knowledge, few studies exploring cross-lingual learning have been done for Portuguese and Spanish, which are close languages and good candidates for that approach. Cruz et al. (2018) presented the first attempt to explore a cross-lingual approach for Portuguese and Spanish, namely direct transfer learning from Spanish to Portuguese. They reported competitive results in comparison with the monolingual model, encouraging further exploration of cross-lingual learning as a way to address co-reference resolution for Portuguese and Spanish.

1.2 Thesis Proposal

This thesis proposes a model for co-reference resolution on Portuguese and Spanish texts. We explore a cross-lingual learning approach, aligning Portuguese and Spanish word embeddings in a single vector space and using data from both languages to train a model. Using cross-lingual learning, our objective is to learn a model capable of efficiently performing co-reference resolution in Portuguese and Spanish, and to compare its results to the ones from the monolingual models.

The proposed co-reference resolution model is an adaptation of a previous state-of-the-art neural system developed for English data - NeuralCoref. We used pre-trained FastText word embeddings to represent the words in the training data. The word embeddings from both languages were later aligned, as mentioned before. Additionally, we also tested some methods for data augmentation, in order to increase the amount of training data, and post-processing the word embeddings, in order to obtain more discriminative representations.

1.3 Contributions

The main contributions of this thesis are as follows:

- The proposal of a neural co-reference resolution model, capable of performing on both Portuguese and Spanish data and achieving promising results. The corresponding code developed during this work is available online[1].
- The evaluation of the system in both a monolingual and a cross-lingual learning scenarios, comparing their results and analysing the effect of cross-lingual learning for Portuguese and Spanish.
- The proposal of a mention detection function, tuned for the Portuguese and Spanish corpora.

1.4 Organization of the Document

This thesis presents the following structure: Chapter 2 introduces the fundamental concepts that serve as basis for this work, and related work on co-reference resolution. Chapter 3 describes the proposed neural network model, its hyperparameters and training strategy, and the cross-lingual word embeddings used. Chapter 4 details the experimental methodology, the datasets and their pre-processing, the evaluation metrics, and the obtained the results. Finally, Chapter 5 presents the main conclusions of this work and suggestions for future work.

[1] https://github.com/NadiaSofia/Co-reference-Resolution-for-PT-and-ES.git

2

Concepts and Related Work

Contents

This chapter presents some background on task-related Machine Learning (ML) concepts, and previous work on co-reference resolution, which are necessary to understand the proposed approach described in a subsequent chapter.

2.1 Fundamental Concepts

This section introduces the fundamental ML concepts that are relevant to this work. We cover Neural Networks, Recurrent Neural Networks, the Transformer architecture, word embeddings, contextual word embeddings and cross-lingual word embeddings.

2.1.1 Learning with Neural Networks

Neural Networks, as the name indicates, are based on the model of the human brain (Goldberg, 2017). Hence, similar to the brain, neural networks are composed by neurons, which act as computation units, and synapses, which act as connections between pairs of neurons. Each connection has an associated weight, similar to the strength of the synapse.

The simplest form of a Neural Network (NN) is called Perceptron and has a single output neuron y. y is computed as the weighted sum of the input neurons $\boldsymbol{x}$, followed by the application of an activation function f (e.g linear and sigmoid functions), as shown in Equation 2.1. $\boldsymbol{w}$ is the weight vector and b is a bias term, added to the weighted sum.

$$y = \mathrm{f}\left(\boldsymbol{x} \cdot \boldsymbol{w} + b\right) \tag{2.1}$$

Adding more layers of neurons to this simple model leads to a Multi-Layer Perceptron (MLP). There are always input and output layers, but the number of hidden layers may vary. Figure 2.1 has an example of a MLP with two hidden layers. Following the Perceptron calculations over the different layers, we get:

$$\begin{aligned} \boldsymbol{h_1} &= \mathrm{f}_1\left(\boldsymbol{x} \cdot \boldsymbol{W_1} + \boldsymbol{b_1}\right) \\ \boldsymbol{h_2} &= \mathrm{f}_2\left(\boldsymbol{h_1} \cdot \boldsymbol{W_2} + \boldsymbol{b_2}\right) \\ \boldsymbol{y} &= \mathrm{f}_3\left(\boldsymbol{h_2} \cdot \boldsymbol{W_3} + \boldsymbol{b_3}\right) \end{aligned} \tag{2.2}$$

In Equation 2.2, $\boldsymbol{W_1}$, $\boldsymbol{W_2}$ and $\boldsymbol{W_3}$ are the weight matrices, $\boldsymbol{x}$ is the input vector, $\boldsymbol{b_1}$, $\boldsymbol{b_2}$ and $\boldsymbol{b_3}$ are the bias vectors, $\boldsymbol{y}$ is the output vector, and f_1, f_2 and f_3 are the activation functions.

Training a NN involves updating its weights and making it possible to predict unseen inputs. When training a NN, loss functions are used to measure how far the predicted output is from the real value, so the objective is to minimize them. There are several loss functions used for classification tasks (e.g. cross-entropy), whose gradient can be easily calculated and used to update the NN weights.

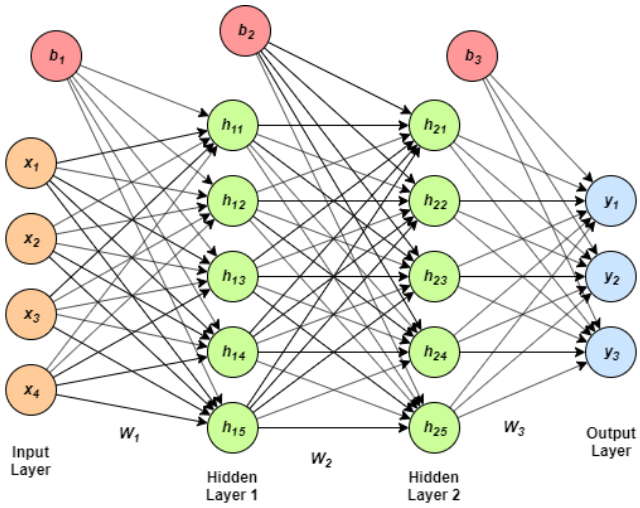

Figure 2.1: Example of a MLP with two hidden layers.

Back-propagation is an algorithm used for computing the gradient of the loss function regarding the network weights, using a chain rule: computing the gradients at the output layer and using those gradients to compute the gradients at the previous layer, and so on. Training a NN thus involves two steps: a forward pass, computing the network outputs given its inputs and calculating the loss; a backward pass, propagating the gradient of the loss function backwards through the network, using it to update the weights. This process is repeated for a defined number of iterations or until convergence.

Using the back-propagation concept, there are some algorithms used to train and optimize NNs, such as the Gradient Descent. The Gradient Descent minimizes the loss function by doing the backward pass and updating the weights of the network in the opposite direction of the gradient of the loss function ($\nabla_\theta L(\theta)$). A learning rate η is used to specify the extent of the update (i.e. the step size). After some iterations, it will converge to a global or local minimum of the loss function. At each step of the Gradient Descent algorithm, the network weights θ can be updated according to the following equation:

$$\theta = \theta - \eta \cdot \nabla_\theta L(\theta) \tag{2.3}$$

There are three variants of Gradient Descent, differing in how much data they use to compute the gradient of the loss function. Batch Gradient Descent computes the gradient for all the training examples, which can be a slow process for large datasets. Stochastic Gradient Descent (SGD), on the other hand, only computes the gradient for one training example. This way the updates are much faster but can have higher variance, causing fluctuation in the loss function. Mini-batch Gradient Descent is a compromise between the two previous variants, lowering the computation time and the variance of the weight updates. For each training iteration, it performs an update for a mini-batch (i.e. a subset) of n training examples.

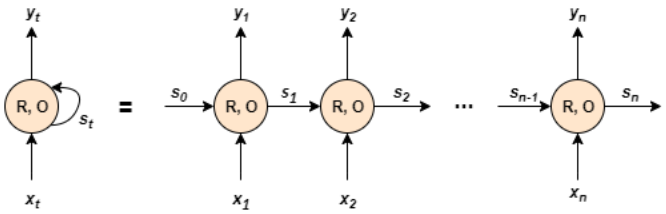

Figure 2.2: Recurrent Neural Network.

The classic SGD algorithm has some known issues, so Ruder (2016) summarizes some optimization algorithms that have been developed over the years. Momentum algorithms, such as Momentum and Nesterov Accelerated Gradient, accumulate the previous gradients and use that for the current update. This increases updates if the gradients point in the same directions and reduces updates if the gradients change directions, leading to a faster convergence. Adaptive Learning Rate algorithms, such as AdaGrad, AdaDelta, RMSProp and Adam, compute an adaptive learning rate per parameter, i.e. use a different learning rate for every parameter and modify (i.e. update) it at every time step.

2.1.2 Recurrent Neural Networks

The order in which words appear in sentences or the order in which sentences appear in a text may be important in Natural Language Processing. Recurrent Neural Networks can be used when there is a sequence to process. They can be abstractly defined as *an interface for translating a sequence of inputs into a fixed sized output* (Goldberg, 2017).

$$\begin{aligned} \mathrm{RNN}\left(\boldsymbol{x_{1:n}}, \boldsymbol{s_0}\right) &= \boldsymbol{y_{1:n}} \\ \boldsymbol{s_i} &= \mathrm{R}\left(\boldsymbol{s_{i-1}}, \boldsymbol{x_i}\right) \\ \boldsymbol{y_i} &= \mathrm{O}\left(\boldsymbol{s_i}\right) \end{aligned} \tag{2.4}$$

Equation 2.4 explains the Recurrent Neural Network (RNN) operation. Given a sequence of vectors $\boldsymbol{x_1}$, ..., $\boldsymbol{x_n}$, the RNN works recursively: it takes a vector $\boldsymbol{x_i}$ and a state vector $\boldsymbol{s_{i-1}}$, and passes them as input to a function R, which returns a new state vector $\boldsymbol{s_i}$. That new state vector $\boldsymbol{s_i}$ is mapped to an output $\boldsymbol{y_i}$ by a function O. The initial state vector, $\boldsymbol{s_0}$, can be given as input to the RNN or defined as a zero vector. The RNN output corresponds to $\boldsymbol{y_{1:n}}$. The structure of a RNN is represented in Figure 2.2.

In the simplest version of a RNN, functions R and O are as follows:

$$\begin{aligned} \mathrm{R}\left(\boldsymbol{s_{i-1}}, \boldsymbol{x_i}\right) &= \mathrm{f}\left(\boldsymbol{s_{i-1}} \cdot \boldsymbol{W^s} + \boldsymbol{x_i} \cdot \boldsymbol{W^x} + \boldsymbol{b}\right) \\ \mathrm{O}\left(\boldsymbol{s_i}\right) &= \boldsymbol{s_i} \end{aligned} \tag{2.5}$$

In Equation 2.5, $\boldsymbol{W^s}$ and $\boldsymbol{W^x}$ are weight matrices, $\boldsymbol{b}$ is a bias vector and f is an activation funtion.

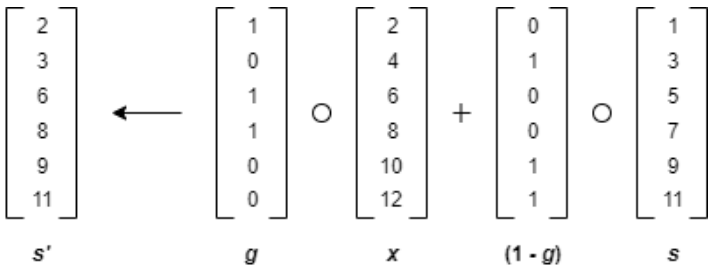

Figure 2.3: Example of using a binary gate vector g to update a state s.

The described RNN architecture allows training a model that can predict a missing word on a sentence, but only based on the previous words. However, the following words can also be useful for the prediction, so there is an elaborated version of a RNN which can take that into account, the Bidirectional Recurrent Neural Network (BiRNN).

Given the input sequence $x_{1:n}$, the BiRNN has two states: a forward state, that receives the input $x_{1:i}$, and a backward state, that receives the input $x_{i:n}$ in reverse (i.e. from x_n to x_i). Both forward and backward states compute the output as a simple RNN. The output of x_i is the concatenation of the outputs from the forward and backward states.

RNNs have two different problems with long-term dependencies: Vanishing Gradient problem and Exploding Gradient problem. In simple RNNs these problems result from repeatedly multiplying the weight matrix W^s for each time step of a long sequence. Vanishing Gradients occur when the weight matrix has small values, making the gradient decrease as it back-propagates, becoming very close to 0. Exploding Gradients are the opposite problem, which occur when the weight matrix has high values, making the gradient increase and become very high.

Gated architectures were created to solve these problems by getting a more controlled memory access. This control is achieved with a binary vector called gate vector (g). When updating to a new state (s'), the entries from the gate vector with 1 correspond to the entries that must be read from the input (x), and the rest should be copied from the current state (s). Figure 2.3 shows the equation for this computation along with an example. The operator $\circ$ is an element wise product.

The gates described still have problems to improve, since they should be dynamic and trainable. These problems can be solved by making the gates differentiable, which means that instead of binary values, they would have real numbers (representing learned weights). This way, the gates become trainable. The real numbers are passed through a sigmoid function, limiting them in the range [0, 1].

An example of this gated architecture is Long Short-Term Memory (LSTM). It works with memory cells, created to preserve memory across time according to a differentiable gated mechanism which determines what to remember and what to forget. LSTM can be defined as shown in Equation 2.6.

$$
\begin{aligned}
\boldsymbol{s_i} = \mathrm{R}_{\mathrm{LSTM}}\left(\boldsymbol{s_{i-1}}, \boldsymbol{x_i}\right) = [\boldsymbol{m_i}; \boldsymbol{h_i}] \\
\boldsymbol{m_i} &= \boldsymbol{f} \circ \boldsymbol{m_{i-1}} + \boldsymbol{i} \circ \boldsymbol{z} \\
\boldsymbol{h_i} &= \boldsymbol{o} \circ \tanh\left(\boldsymbol{m_i}\right) \\
\boldsymbol{i} &= \sigma(\boldsymbol{x_i} \cdot \boldsymbol{W_{x,i}} + \boldsymbol{h_{i-1}} \cdot \boldsymbol{W_{h,i}}) \\
\boldsymbol{f} &= \sigma(\boldsymbol{x_i} \cdot \boldsymbol{W_{x,f}} + \boldsymbol{h_{i-1}} \cdot \boldsymbol{W_{h,f}}) \\
\boldsymbol{o} &= \sigma(\boldsymbol{x_i} \cdot \boldsymbol{W_{x,o}} + \boldsymbol{h_{i-1}} \cdot \boldsymbol{W_{h,o}}) \\
\boldsymbol{z} &= \tanh\left(\boldsymbol{x_i} \cdot \boldsymbol{W_{x,z}} + \boldsymbol{h_{i-1}} \cdot \boldsymbol{W_{h,z}}\right) \\
\boldsymbol{y_i} = \mathrm{O}_{\mathrm{LSTM}}\left(\boldsymbol{s_i}\right) = \boldsymbol{h_i}
\end{aligned}
\tag{2.6}
$$

The new state $\boldsymbol{s_i}$ is a composition of two vectors: the first one is the memory cell $\boldsymbol{m_i}$, and the second one is the hidden state $\boldsymbol{h_i}$. There are three gates: an input gate $\boldsymbol{i}$, deciding which parts of the input to write to the new memory state, a forget gate $\boldsymbol{f}$, deciding which parts from the previous memory to forget, and an output gate $\boldsymbol{o}$, deciding what should be passed to the output. The three gates are updated based on the input $\boldsymbol{x_i}$, the previous hidden state $\boldsymbol{h_{i-1}}$, and the learned weights ($\boldsymbol{W_{x,i}}$, $\boldsymbol{W_{h,i}}$, $\boldsymbol{W_{x,f}}$, $\boldsymbol{W_{h,f}}$, $\boldsymbol{W_{x,o}}$, $\boldsymbol{W_{h,o}}$), with the results passed through a sigmoid function (σ). Vector $\boldsymbol{z}$ corresponds to the updated values, based on a linear combination of the input $\boldsymbol{x_i}$ and the previous hidden state $\boldsymbol{h_{i-1}}$ involving weights ($\boldsymbol{W_{x,z}}$, $\boldsymbol{W_{h,z}}$), and passed through a $\tanh$ activation function. The forget and input gates are used to update the memory cell $\boldsymbol{m_i}$, deciding what to pass from the previous memory $\boldsymbol{m_{i-1}}$ and the updated values $\boldsymbol{z}$, respectively. The hidden state $\boldsymbol{h_i}$, which is also the output $\boldsymbol{y_i}$, is determined based on the output gate $\boldsymbol{o}$ and the computed memory component $\boldsymbol{m_i}$ (passed through a $\tanh$ function).

2.1.3 The Transformer Architecture

In this section we will cover the Transformer architecture, presented by Vaswani et al. (2017). Transformer is the first transduction model computing the input and output representations by relying entirely on an attention mechanism, containing no recurrence and no convolution.

The Transformer has two components: an encoder and a decoder, both composed of a stack of six identical layers. The encoder maps the input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\boldsymbol{z} = (z_1, ..., z_n)$. These continuous representations are used by the decoder to generate an output sequence of symbols $(y_1, ..., y_m)$, one at a time. At each step, the previously generated symbols are used as additional input to generate the next ones. The Transformer architecture is shown in Figure 2.4.

Each layer (i.e. encoder) in the encoder stack has two sub-layers: one implementing a multi-head attention mechanism, followed by another one implementing a Feed-Forward Neural Network (FFNN). The attention sub-layer helps the encoder look at other words (i.e. positions) in the input as it encodes a

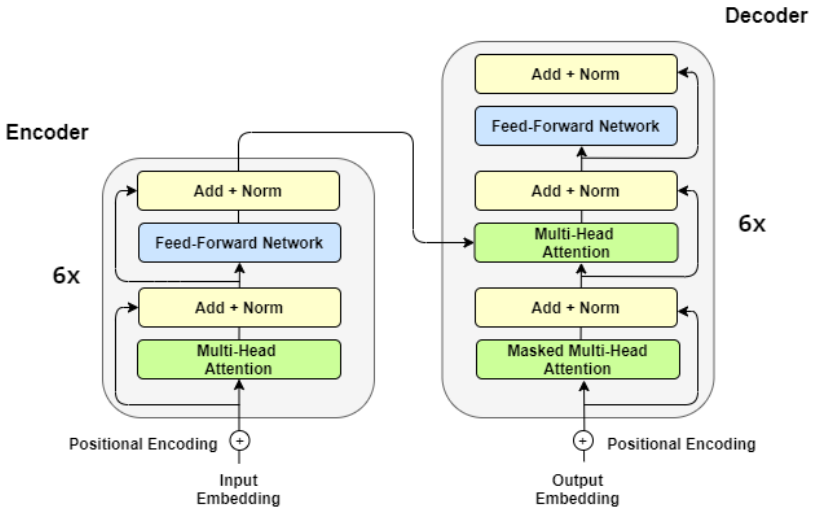

Figure 2.4: Transformer architecture.

specific one, by attending the previous layer. There is a residual connection (skips nonlinear processing) around each sub-layer, followed by layer normalization - the values from the residual connection are added to the output of the function implemented by the sub-layer, and then normalized.

Each layer (i.e. decoder) in the decoder stack also has the two mentioned sub-layers, plus an additional one between them implementing multi-head attention over the encoder stack output. The bottom attention sub-layer is also modified to prevent a position from attending to subsequent ones, which is done by masking them. Similar to the encoder, there are residual connections around each sub-layer, followed by normalization.

The concept of attention was used in encoder and decoder layers, so let us explain it. Vaswani et al. (2017) define attention as a function that maps a query and a set of key-value pairs to an output, having the output computed as a weighted sum of the values. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Vaswani et al. (2017) call their attention *Scaled Dot-Product Attention*, whose output is computed as shown in the following equation:

$$\text{Attention}\,(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\,(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{d_k}}) \cdot \boldsymbol{V} \tag{2.7}$$

In Equation 2.7, $\boldsymbol{Q}$ is the matrix of queries and $\boldsymbol{K}$ is the matrix of keys, both with dimension d_k. $\boldsymbol{V}$ is the matrix of values, with dimension d_v. They compute the dot product of the queries ($\boldsymbol{Q}$) with the keys ($\boldsymbol{K}$), multiply it by $\frac{1}{\sqrt{d_k}}$ (a scaling factor) and apply a softmax function. That component represents the weights to apply on the values.

The dot products grow large in magnitude for large values of d_k, pushing the softmax function into regions with extremely small gradients, so the scaling factor is used to counteract that.

Instead of using just one attention function, the multi-head attention implements parallel functions of attention, performed on linear projections of the queries, keys and values. The values obtained from the attention functions are concatenated and projected again, as shown in Equation 2.8, obtaining the final output values. The matrices $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$, $\boldsymbol{W}_i^V$ and $\boldsymbol{W}^O$ represent the projections.

$$\begin{aligned} \text{MultiHead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) &= \text{Concat}\left(head_1, ..., head_h\right)\boldsymbol{W}^O \\ head_i &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right) \end{aligned} \tag{2.8}$$

As explained in the beginning, there is no recurrence and no convolution in a Transformer, so to get the model to use tokens' order in the sequence, some information about the token position must be injected. This task can be approached using *positional encodings*, added to the input embeddings in the beginning of the encoder and decoder stacks.

2.1.4 Representing Text with Word Embeddings

Computational analysis on a text cannot be done using the words themselves, since a literal comparison between two words (i.e. character by character) is not a good way to see if they are related. For example, blue and white are related since they are both colors, but they have no type of spelling relation. As another flaw, the same word can have different meanings, misleading into an incorrect similarity (e.g. duck, as a noun or a verb). Considering that, one of the first tasks in text processing is representing the text in a mathematical way. The typical approach is getting a model that converts the text to a vector, and the goal is that similar words have close representations. Relations like synonymy, antonymy, hyponymy, hyperonymy, holonymy and meronymy are good indicators of word similarity. However, representing all of these relations for a huge amount of words would be a very long and difficult task. Another way of recognizing words similarity is from their context, by analysing different sentences or corpora with the words in it. In this subsection we will present some approaches used for representing text as a vector, and in the next subsection we will present approaches including contextual information in the representations.

One-hot representation is the most basic approach for text representation. According to Smith (2019), it consists in giving each word (i.e. feature) its own dimension and assigning it 1 (all the other dimensions for that word are 0). Since there is one dimension per word, the vectors would have a high dimensionality. Also, this representation does not allow recognizing similarity between the words.

Word embeddings are an alternative representation, with much lower dimensionality. Unlike one-hot representation, they allow recognizing similarity between words. Word embeddings can be obtained through a supervised or an unsupervised pre-training. Supervised pre-training is used when there is a lot of labeled data available for comparison regarding a specific task. In these cases, the vectors can be trained using a NN, obtaining good representations. When there is not enough labeled data available, which is the most common case, unsupervised pre-training is used on lots of unlabeled data. In these

cases, the vectors are obtained following the idea that similar words appear in similar contexts. Given this, we can have models that try to predict a missing word from its context or try to predict the context from the word. We will present two of those word embedding methods: Word2Vec and FastText.

Word2Vec uses a NN with one hidden layer that encodes the word embedding representation. As explained before, the words around the target word (i.e. its context) are used to obtain the representation. There are two variants of Word2Vec: Skip-Gram and Continuous Bag of Words (CBOW). Skip-Gram is used to predict the words surrounding a given target word. Hence, the input corresponds to the target word, and the output are the context words (according to a context window, C). The hidden layer contains the embedding representation for the target word (can be extracted from there). The input and output vectors are one-hot encoded, having the same dimensionality (for W words, W and $C*W$, respectively). Only the embedding representations are lower sized vectors. A softmax activation function is used in the output to compute the probability for each word to appear at the context window places. CBOW is similar to Skip-Gram, but for the opposite prediction, as it is used to predict a target word given a context. The performances of these two variants are similar in terms of efficiency, even though Skip-Gram works a little better with rare words.

FastText is similar to Word2Vec, but instead of using complete words it uses n-grams (i.e. contiguous parts of the word with size n). For example, bi-grams for *home* would be *ho*, *om* and *me*. In this case, the word embedding corresponds to the sum of all the n-grams representations. According to this model, words that share n-grams also share part of their representation. The advantage of FastText when compared to Word2Vec is that it allows representing rare words easily and better, since their n-grams probably appeared in other words (hence, are recognized).

2.1.5 Contextual Word Embeddings

The most recent approaches of word embeddings are based on representing a word according to its context. In this section we will present Embeddings from Language Models (ELMo) and Bidirectional Encoder Representations from Transformers (BERT), two contextual word embedding models.

ELMo uses the context from both sides of the target word to represent it, training a Bidirectional Language Model (BiLM). We will start by explaining the operation of a BiLM. According to Peters et al. (2018), for a sequence of N words, BiLM *combines a forward and backward language model to jointly maximize the log likelihood of both directions*:

$$\sum_{n=1}^{N} [\log p(t_n|t_1, ..., t_{n-1}; \Theta_F) + \log p(t_n|t_{n+1}, ..., t_N; \Theta_B)] \tag{2.9}$$

In Equation 2.9, the first part of the sum represents the forward operation: given the sequence of

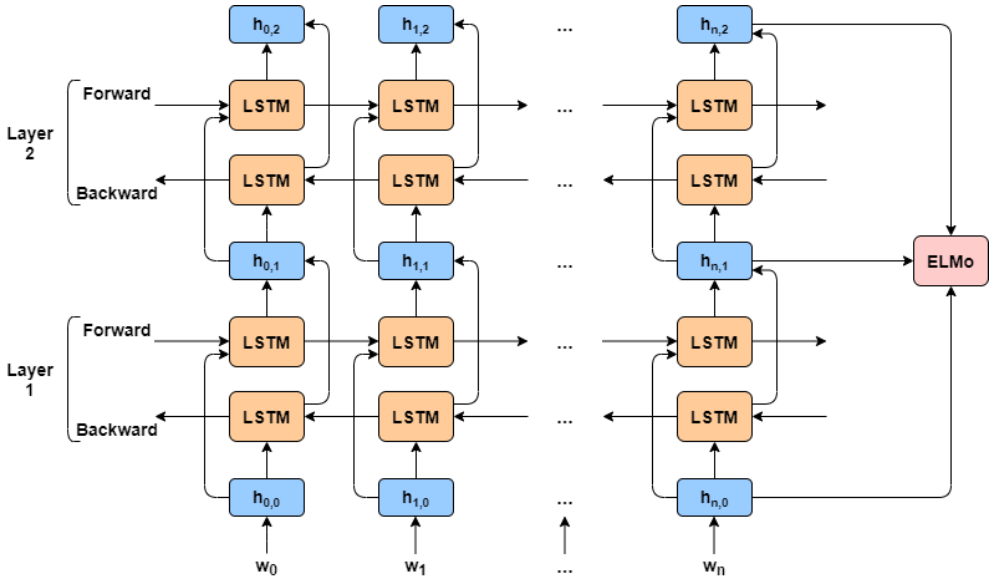

Figure 2.5: ELMo architecture for two layers.

previous $n-1$ words, it shows the probability of the token t_n being the n^{th} word. The second part of the sum represents the backward operation: given the sequence of following words (from $n+1$ to N), it shows the probability of the token t_n being the n^{th} word. Θ_F and Θ_B are the parameters for the forward and backward language models, respectively.

The probability of the next token is calculated in two steps: first getting a word representation $\boldsymbol{x_n}$ without the context information, and second computing L layers of context-dependent representations $\overrightarrow{\boldsymbol{h}}_{n,i}$, $i \in [1, L]$. The upper layer output $\overrightarrow{\boldsymbol{h}}_{n,L}$ is used to predict the following token using a softmax function. This process is done for forward and backward language models, and their results are concatenated, forming the L layers of contextual representations per token position: $\boldsymbol{h_{n,i}} = [\overrightarrow{\boldsymbol{h}}_{n,i}, \overleftarrow{\boldsymbol{h}}_{n,i}]$.

The language model used by ELMo is LSTM (previously described in Subsection 2.1.2), having *L* Bidirectional Long Short-Term Memory (BiLSTM) layers. ELMo representations are obtained by multiplying all layers by weights and combining them, as shown in the following equation:

$$\mathrm{ELMo_n} = \lambda \sum_{l=0}^{L} \boldsymbol{s_l} \cdot \boldsymbol{h_{n,l}} \tag{2.10}$$

In Equation 2.10, $\boldsymbol{s_l}$ are the weights and λ is a scalar number. Since ELMo is task specific, both of these parameters are also specific according to the task. Without the specific parameters, ELMo could just take the upper layer information. ELMo architeture for $L = 2$ layers is represented in Figure 2.5.

ELMo can only analyze the forward and backward contexts separately, concatenating the results afterwards. BERT, introduced by Devlin et al. (2018), is an improved contextual word embedding model that intends to look at the right and left context as a whole, avoiding the unidirectionality constraint.

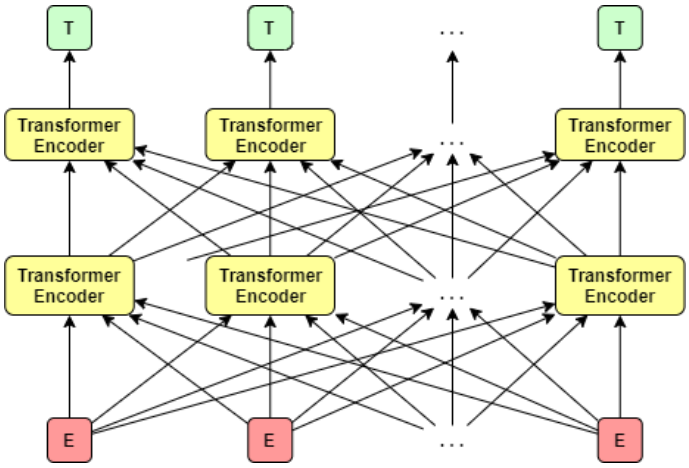

Figure 2.6: BERT architecture.

Focusing on the architecture, BERT is a multi-layer bidirectional Transformer encoder based on the work of Vaswani et al. (2017). Figure 2.6 shows BERT architecture. The first layer has the input embedding (E), followed by two layers with the Transformer encoders and lastly the layer having the contextual representations of each token (T). The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. Token embeddings represent each word, position embeddings have information about the relative position of the tokens in the sentence, and segmentation embeddings have information about sentences separation and which sentence a token belongs to.

BERT is divided in two steps: pre-training and fine-tuning. Pre-training consists in training the model on unlabeled data for different tasks. The pre-trained parameters obtained are used to initialize the model and then they are fine-tuned using labeled data from the downstream tasks, representing the fine-tuning step.

Getting more into the pre-training step, BERT does not use different models to analyze forward or backward information. Instead, it uses two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction. MLM task is what allows BERT to be bidirectional. It masks a small percentage of the input tokens (i.e. replaces them by a [MASK] token), chosen randomly, and then predicts them. This allows a right and left joint analysis for the model training. During the training process, the small percentage of tokens chosen is not always masked: 80% of the times they are effectively masked, 10% of the times they are replaced with a random token, 10% of the times they remain unchanged. The second task of pre-training, as the name says, predicts the next sentence. The model is pre-trained on the basis that, having two sentences S_1 and S_2, S_2 follows S_1 50% of the times and the remaining 50% of the times there is another sentence following S_1.

2.1.6 Cross-Lingual Word Embeddings

Cross-lingual approaches to word embeddings attempt to unify language representations, trying to get similar representations according to the words' meaning regardless of their language.

According to Ruder et al. (2017), cross-lingual word embedding methods differ regarding two dimensions: the type of alignment and the comparability between data sources in the different languages. The type of alignment can be at the level of words, sentences or documents, referring to which level the translation between texts is made. Word alignments refer to pairs of translations between words in different languages, for example from bilingual dictionaries. Sentence alignments refer to a parallel corpus, containing parallel sentences in the source and target languages that are translations of each other. Document alignment refers to documents in different languages that are translations of each other. Regarding the comparability, data sources can be parallel, i.e. the exact translation in different languages, or comparable, if the data is only similar in some way (e.g. share the same topic).

We will present some word alignment methods out of which part of them require parallel data and part of them do not.

Mikolov et al. (2013) observed that word vector representations from different languages have similar geometric arrangements in both vector spaces. This suggests that it is possible to transform the vector space of a source language to the vector space of a target language by learning a linear mapping, with a transformation matrix $\boldsymbol{W}$. To obtain that transformation, they start by independently learning the word vector representations (using Skip-Gram or CBOW models) of each language using large monolingual corpora. Then, they use the n most frequent words in the source language and their translations in the target language as seed words, obtaining a bilingual dictionary. Having the word pairs from the dictionary and their associated vector representations $\{\boldsymbol{x_i}, \boldsymbol{z_i}\}_{i=1}^{n}$, where $\boldsymbol{x_i}$ is the representation of word i in the source language and $\boldsymbol{z_i}$ is the representation of its translation, the objective is to find the transformation $\boldsymbol{W}$ such that $\boldsymbol{W}\boldsymbol{x_i}$ approximates $\boldsymbol{z_i}$. For such, $\boldsymbol{W}$ is learned by minimizing the squared Euclidean distance between $\boldsymbol{W}\boldsymbol{x_i}$ and $\boldsymbol{z_i}$, as shown in Equation 2.11, which is solved using SGD.

$$\arg\min_{\boldsymbol{W}} \sum_{i=1}^{n} ||\boldsymbol{W}\boldsymbol{x_i} - \boldsymbol{z_i}||^2 \tag{2.11}$$

With this, any given new word in the source language and its vector representation $\boldsymbol{x_{new}}$ can be mapped to the target language space by computing $\boldsymbol{z_{new}} = \boldsymbol{W}\boldsymbol{x_{new}}$.

In the method proposed by Mikolov et al. (2013), the number of word pairs in the bilingual dictionary must be high, in the order of few thousand entries. Since dictionaries of that size are difficult to obtain for many language pairs, Artetxe et al. (2017) proposed a method to reduce the bilingual resources needed, working with as little as a 25 word pairs dictionary. Similar to the previous method, they train the embeddings for each language on monolingual corpora, and then learn a linear transformation that

maps the embeddings from one language space to the other. They proposed a self-learning approach to learn the transformation that can be combined with any dictionary-based mapping technique. Existent dictionary-based mapping techniques use the seed dictionary to learn the embedding mapping $\boldsymbol{W}$ and then use the embedding mapping to induce a new dictionary. The self-learning method iteratively repeats that process, using the induced dictionary from the previous mapping as input to learn the new mapping, until some convergence criterion is met. This iterative method gradually aligns embedding spaces.

Both previous methods require parallel data. Conneau et al. (2017) proposed a model to align monolingual word embedding spaces in an unsupervised way (i.e. without parallel data). The method is divided in three steps: adversarial learning, refinement and cross-domain similarity local scaling. The code, embeddings and dictionaries for this method are publicly available in the MUSE library[1].

Having $X = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$ and $Y = \{\boldsymbol{y_1}, ..., \boldsymbol{y_m}\}$ as two sets of n and m word embeddings trained independently on monolingual data from source and target languages, respectively, Conneau et al. (2017) uses adversarial training to learn a mapping $\boldsymbol{W}$ from the source to the target space. The adversarial learning consists of training a model - the discriminator - to differentiate between embeddings sampled from $\boldsymbol{W}X$ and Y, while training the mapping $\boldsymbol{W}$ to make $\boldsymbol{W}X$ and Y as similar as possible. This way, the discriminator aims at maximizing its ability to identify the origin of an embedding (source or target language), and the mapping $\boldsymbol{W}$ aims at preventing the discriminator from making accurate predictions. The discriminator and mapping losses are described in Equations 2.12 and 2.13. For every input sample, the discriminator and the mapping $\boldsymbol{W}$ are trained successively with stochastic gradient updates to minimize their losses.

$$\mathrm{L_D} = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\mathrm{source} = 1|\boldsymbol{W}\boldsymbol{x_i}) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\mathrm{source} = 0|\boldsymbol{y_i}) \tag{2.12}$$

$$\mathrm{L_W} = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\mathrm{source} = 0|\boldsymbol{W}\boldsymbol{x_i}) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\mathrm{source} = 1|\boldsymbol{y_i}) \tag{2.13}$$

In Equations 2.12 and 2.13, θ_D are the discriminator parameters and $P_{\theta_D}(\mathrm{source} = 1|\boldsymbol{z})$ is the probability of a vector $\boldsymbol{z}$ being the mapping of a source embedding (i.e., belonging to the source language). The implemented discriminator is a MLP with two hidden layers and Leaky-ReLU activation functions.

To refine the obtained mapping $\boldsymbol{W}$, Conneau et al. (2017) select the most frequent words and build a dictionary with them, retaining only mutual nearest neighbors (i.e. the pairs of vectors $(\boldsymbol{x}, \boldsymbol{y})$ for which $\boldsymbol{x}$ is the nearest neighbor of $\boldsymbol{y}$ and vice-versa). Then the Procrustes solution, which uses the Singular Value Decomposition (SVD) of $\boldsymbol{Y}\boldsymbol{X}^{\boldsymbol{T}}$, is applied on the created dictionary, as shown in Equation 2.14. d is the number of entries in the created dictionary and $\boldsymbol{W}^*$ is the improved mapping obtained.

[1] https://github.com/facebookresearch/MUSE

$$
\begin{aligned}
\boldsymbol{W}^* &= \arg\min_{\boldsymbol{W}} \sum_{i=1}^{d} ||\boldsymbol{W}\boldsymbol{x_i} - \boldsymbol{y_i}||^2 \\
&= \arg\min_{\boldsymbol{W}} ||\boldsymbol{W}\boldsymbol{X} - \boldsymbol{Y}||_{\mathrm{F}}^2 \\
&= \boldsymbol{U}\boldsymbol{V}^{\boldsymbol{T}} \text{ , with } \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} = \mathrm{SVD}(\boldsymbol{Y}\boldsymbol{X}^{\boldsymbol{T}})
\end{aligned}
\tag{2.14}
$$

In order to produce reliable matching pairs between two languages such that the words per pair are mutual nearest neighbors, Conneau et al. (2017) apply a similarity measure called Cross-Domain Similarity Local Scaling (CSLS), which expands the spaces with high density of points, increasing the distance between word vectors. Considering a mapped source embedding $\boldsymbol{W}\boldsymbol{x_s}$ and its neighborhood $N_T(\boldsymbol{W}\boldsymbol{x_s})$ in the target language, as well as a target embedding $\boldsymbol{y_t}$ and its neighborhood $N_S(\boldsymbol{y_t})$ in the source language, the similarity measure CSLS between mapped source words and target words is computed as shown in Equation 2.15. K is the number of nearest neighbors in each set, $\cos(\cdot,\cdot)$ is the cosine similarity, r_T denotes the mean similarity of a source word to its target neighborhood and, likewise, r_S denotes the mean similarity of a target word to its source neighborhood. CSLS increases the similarity of isolated word vectors and decreases the similarity of vectors in dense areas.

$$
\begin{aligned}
\mathrm{CSLS}(\boldsymbol{W}\boldsymbol{x_s}, \boldsymbol{y_t}) &= 2\cos(\boldsymbol{W}\boldsymbol{x_s}, \boldsymbol{y_t}) - r_T(\boldsymbol{W}\boldsymbol{x_s}) - r_S(\boldsymbol{y_t}) \\
r_T(\boldsymbol{W}\boldsymbol{x_s}) &= \frac{1}{K} \sum_{\boldsymbol{y_t} \in N_T(\boldsymbol{W}\boldsymbol{x_s})} \cos(\boldsymbol{W}\boldsymbol{x_s}, \boldsymbol{y_t}) \\
r_S(\boldsymbol{y_t}) &= \frac{1}{K} \sum_{\boldsymbol{W}\boldsymbol{x_s} \in N_S(\boldsymbol{y_t})} \cos(\boldsymbol{W}\boldsymbol{x_s}, \boldsymbol{y_t})
\end{aligned}
\tag{2.15}
$$

Schuster et al. (2019) build upon ELMo to align contextual word embeddings from a source language to a target language. Instead of learning the alignment in the contextual space itself, they define a context-independent embedding anchor for each token as the average of its different contextual embeddings, and use it as a unique representation of the token in the mapping process. They proposed two alignment methods, one supervised and another unsupervised. For the supervised alignment, they use a dictionary for source and target domains, and embedding anchors for the words in the source and target languages. In this case, the alignment matrix $\boldsymbol{W}$ is obtained with the method proposed by Mikolov et al. (2013) (previously explained in Equation 2.11), using the embedding anchors as the word representations. For the unsupervised alignment, they use only the embedding anchors, and apply the adversarial algorithm and the refinement procedure in the MUSE library (Conneau et al., 2017) to obtain the alignment. To create the dictionary, they use distance in the anchor space.

2.2 Related Work

This section contains previous work developed on co-reference resolution, divided in two subsections: Subsection 2.2.1, exposing some state-of-the-art models, specifically a model related to our baseline and an end-to-end system; Subsection 2.2.2, presenting work focused on Portuguese and Spanish texts, including a model exploring direct transfer learning from Spanish to Portuguese.

2.2.1 State-of-the-art Models for Co-Reference Resolution

Over the years, several works have been developed for co-reference resolution (Sukthanker et al., 2018). Initial approaches worked on rule based resolution, using hand-crafted rules based on syntactic and semantic features of the text. Over the years, co-reference resolution shifted to ML approaches (e.g. using decision trees) and recently to deep learning models, relying on NNs. The latter models present the best results on the task and correspond to the state-of-the-art. Stylianou and Vlahavas (2019) presented a review on several neural models, such as the ones developed by Clark and Manning (2016) and Lee et al. (2017), achieving high results on the English CoNLL-2012 dataset.

Clark and Manning (2016) presented an entity-level approach where they use clusters of mentions to capture global information, using a cluster-ranking model to decide which clusters to merge. Each mention starts as a singleton cluster and then, during inference, the pairs of clusters considered to be referring to the same entity are merged. The system includes four different components: a mention-pair encoder, a cluster-pair encoder, a mention-ranking model and a cluster-ranking model. The first three components work together to feed the cluster-ranking model, which makes the merging decisions.

The mention-pair encoder produces a distributed representation $\boldsymbol{r_m}$ for pairs of mentions by passing relevant features through a FFNN. A pair of mentions (a, m) contains a mention m and a candidate antecedent a, which might be a mention that occurs before m in the document or NA, indicating that m has no antecedent. In the input layer, the model extracts several features regarding the mention, the candidate antecedent, the pair and the document, concatenating their vectors to produce the input vector $\boldsymbol{h_0}$. The set of features includes: embedding features for m and a (e.g. word embeddings of the head word and first word); additional mention features for m and a (e.g. type of mention, mention position); document genre (e.g. broadcast news, web data); distance features (e.g. the distance between m and a in sentences); speaker features (e.g. whether m and a have the same speaker); string matching features (e.g. exact string match, partial string match). If $a = \mathrm{NA}$, the features regarding mention pairs are not included. The input $\boldsymbol{h_0}$ gets passed through a three-layer fully-connected FFNN, with ReLU activation functions, as shown in Equation 2.16. The output of the last hidden layer is the vector representation for the mention pair: $\boldsymbol{r_m}(a, m) = \boldsymbol{h_3}(a, m)$.

$$\boldsymbol{h_i}(a, m) = \max(0, \boldsymbol{W_i h_{i-1}(a, m)} + \boldsymbol{b_i}) \tag{2.16}$$

The cluster-pair encoder produces a distributed representation $\boldsymbol{r_c}$ for a pair of clusters of mentions (c_i, c_j). The encoder applies max-pooling and average-pooling (column-wise) to the matrix of mention-pair representations $\boldsymbol{R_m}(c_i, c_j)$, which includes all mention-pairs between clusters c_i and c_j, and concatenates the results, as shown in Equation 2.17. d is the dimension of the mention-pair representations.

$$\boldsymbol{r_c}(c_i, c_j)_k = \begin{cases} \max\{\boldsymbol{R_m}(c_i, c_j)_{k,\cdot}\} & \text{for } 0 \leq k \leq d \\ \text{avg}\{\boldsymbol{R_m}(c_i, c_j)_{k-d,\cdot}\} & \text{for } d \leq k \leq 2d \end{cases} \tag{2.17}$$

The mention-ranking model assigns a score s_m for each mention-pair produced by the mention-pair encoder, representing their compatibility for co-reference. This is achieved by applying a single fully-connected layer to the representation produced by the mention-pair encoder. The mention-ranking model is trained using a slack-rescaled max-margin loss function:

$$\begin{aligned} \mathrm{L}_{\text{Ranking}} &= \sum_{i=1}^{N} \max_{a \in A(m_i)} \delta(a, m_i)(1 + s_m(a, m_i) - s_m(\hat{t}_i, m_i)) \\ \hat{t}_i &= \arg\max_{t \in T(m_i)} s_m(t, m_i) \end{aligned} \tag{2.18}$$

In Equation 2.18, N is the number of mentions in the training set, $A(m_i)$ is the set of candidate antecedents of mention m_i (i.e. preceding mentions and NA), $T(m_i)$ is the set of true antecedents of m_i and $\hat{t}_i$ is the highest scoring true antecedent of m_i. $\delta(a, m_i)$ is a mistake-specific cost function for False New (FN), False Anaphoric (FA), Wrong Link (WR) and correct co-reference decisions, defined as:

$$\delta(a, m_i) = \begin{cases} \alpha_{FN} & \text{if } a = \text{NA} \wedge T(m_i) \neq \{\text{NA}\} \\ \alpha_{FA} & \text{if } a \neq \text{NA} \wedge T(m_i) = \{\text{NA}\} \\ \alpha_{WL} & \text{if } a \neq \text{NA} \wedge a \notin T(m_i) \\ 0 & \text{if } a \in T(m_i) \end{cases} \tag{2.19}$$

The pre-training of the network is done in two stages, minimizing two objectives: All-Pairs Classification and Top-Pairs Classification, represented in Equations 2.20 and 2.21, respectively. $F(m_i)$ if the set of false antecedents of m_i and $p(a, m_i) = \sigma(s_m(a, m_i))$. They are both cross-entropy losses, so All-Pairs is optimized by maximizing scores of mention-pairs truly co-referent and minimizing the others, and Top-Pairs is optimized in a similar way but focusing on the high-scoring mention-pairs.

$$\mathrm{L}_{\text{All-Pairs}} = -\sum_{i=1}^{N} \left[\sum_{t \in T(m_i)} \log p(t, m_i) + \sum_{f \in F(m_i)} \log(1 - p(f, m_i)) \right] \tag{2.20}$$

$$\mathrm{L}_{\text{Top-Pairs}} = -\sum_{i=1}^{N} \left[\max_{t \in T(m_i)} \log p(t, m_i) + \min_{f \in F(m_i)} \log(1 - p(f, m_i)) \right] \tag{2.21}$$

Finally, the cluster-ranking model obtains a score $s_c(c_i, c_j)$ for a pair of clusters (c_i, c_j), representing their compatibility for co-reference, by applying a single fully-connected layer to the representation $r_c(c_i, c_j)$ produced by the cluster-pair encoder. It also measures a score of anaphoricity $s_{\mathrm{NA}}(m)$ for a mention m, using the representation $\boldsymbol{r_m}(\mathrm{NA}, m)$ from the mention-pair encoder.

In the beginning, each cluster contains a single mention. Let c_m be the cluster containing mention m and c be a cluster containing a mention in the set of candidate antecedents of m. The decision to combine or not the pair of clusters (c_m, c) is made by a policy network which takes into account the cluster-ranking and anaphoricity scores to decide whether to merge (if $s_c(c_m, c)$ is the highest) or pass (if $s_{\mathrm{NA}}(m)$ is the highest). The order in which to consider the mentions is defined by sorting them in decreasing order by their highest candidate co-reference link score, which is computed as $s_m(a, m) - s_m(\mathrm{NA}, m)$ for candidate a and mention m. With this order, the easy decisions are made first. The set of candidate antecedents for each mention is also pruned, including only the ones with high co-reference link scores.

As future decisions are based on previous ones, the cluster-ranking model is trained using a learn-to-search algorithm, which projects all possible actions taken by the policy network. The algorithm assigns a cost to each possible action by executing the action and *rolling out* from the result following a reference policy until the end, computing the resulting loss. According to Clark and Manning (2016), this *rolling out* procedure allows the model to learn how a local action will affect the final score. The cost assigned to the action corresponds the resulting loss, which for this case is defined as the negative of the B^3 co-reference metric. The reference policy takes the action that increases the B^3 score the most at each step. The policy network is trained to minimize the risk associated with taking each action.

Lee et al. (2017) presented the first end-to-end neural co-reference resolution model. The model considers all spans in a document as possible mentions and learns which of them are entity mentions, along with how to best cluster them. The model assigns an antecedent to each span, which can be one of the preceding spans or a dummy antecedent. The dummy antecedent (ϵ) covers two possibilities: the span not being an entity mention or the span being an entity mention but not co-referent with any previous span. For a span i and an antecedent y_i from the set $Y(i) = \{\epsilon, 1, ..., i-1\}$, the model computes the following conditional probability:

$$P(y_i|D) = \frac{\exp\left(\mathrm{s}(i, y_i)\right)}{\sum_{y' \in Y(i)} \exp\left(\mathrm{s}(i, y')\right)} \tag{2.22}$$

In Equation 2.22, D is the document and $\mathrm{s}(i, y)$ is a pairwise score for a co-reference link between spans i and y, depending on three factors: span i being a mention, span y being a mention and y being an antecedent of i. Following this, $\mathrm{s}(i, y)$ is computed as:

$$\mathrm{s}(i,y) = \begin{cases} 0, & y = \epsilon \\ \mathrm{s_m}(i) + \mathrm{s_m}(y) + \mathrm{s_a}(i,y), & y \neq \epsilon \end{cases} \tag{2.23}$$

In Equation 2.23, $\mathrm{s_m}(i)$ is a unary score for span i being a mention and $\mathrm{s_a}(i,y)$ is a pairwise score for span y being an antecedent of span i. Fixing the score of the dummy antecedent to 0, the best scoring antecedent is predicted if any other antecedent score is positive, otherwise the model abstains. The scores are computed via standard FFNNs:

$$\begin{aligned} \mathrm{s_m}(i) &= \boldsymbol{w_m} \cdot \mathrm{FFNN_m}(\boldsymbol{g_i}) \\ \mathrm{s_a}(i,y) &= \boldsymbol{w_a} \cdot \mathrm{FFNN_a}([\boldsymbol{g_i}, \boldsymbol{g_y}, \boldsymbol{g_i} \circ \boldsymbol{g_y}, \phi(i,y)]) \end{aligned} \tag{2.24}$$

In Equation 2.24, $\boldsymbol{g_i}$ is the vector representation for span i, $\boldsymbol{w_m}$ and $\boldsymbol{w_a}$ are weights, $\boldsymbol{g_i} \circ \boldsymbol{g_y}$ is an element-wise multiplication between spans and $\phi(i,y)$ is a feature vector encoding speaker and genre information, and the distance between the two spans.

To obtain the span representations ($\boldsymbol{g_i}$) Lee et al. (2017) use a BiLSTM, to encode lexical information of the inside and outside of the span, along with an attention mechanism over the words in each span, to model a head word vector. The head-finding attention mechanism works as follows:

$$\begin{aligned} \hat{\boldsymbol{x}}_{\boldsymbol{i}} &= \sum_{t=\mathrm{START(i)}}^{\mathrm{END(i)}} a_{i,t} \cdot \boldsymbol{x_t} \\ a_{i,t} &= \frac{\exp(\alpha_t)}{\sum_{k=\mathrm{START(i)}}^{\mathrm{END(i)}} \exp(\alpha_k)} \\ \alpha_t &= \boldsymbol{w_\alpha} \cdot \mathrm{FFNN}_\alpha(\boldsymbol{x_t^*}) \end{aligned} \tag{2.25}$$

In Equation 2.25, $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ (i.e. the head vector) is a weighted sum of word vectors in span i ($\boldsymbol{x_t}$), $a_{i,t}$ are weights and $\boldsymbol{x_t^*}$ is the concatenated output of the BiLSTM. The final span representation is a concatenation of the span boundary representations from BiLSTM ($\boldsymbol{x^*_{START(i)}}$ and $\boldsymbol{x^*_{END(i)}}$), the head vector ($\hat{\boldsymbol{x}}_{\boldsymbol{i}}$) and a feature vector encoding the size of the span ($\phi(i)$):

$$\boldsymbol{g_i} = [\boldsymbol{x^*_{START(i)}}, \boldsymbol{x^*_{END(i)}}, \hat{\boldsymbol{x}}_{\boldsymbol{i}}, \phi(i)] \tag{2.26}$$

Figure 2.7 shows the computation of span representations and co-reference scores in Lee et al. (2017) model. Since the size of the model is too big, the candidate spans are greedily pruned during training and evaluation, according to its mention scores. Only a certain number of spans with the highest mention scores are considered, as only a maximum number of antecedents is considered for each.

Since in the training data only clustering information is observed, the model optimizes the marginal log-likelihood of all correct antecedents implied by the gold clustering:

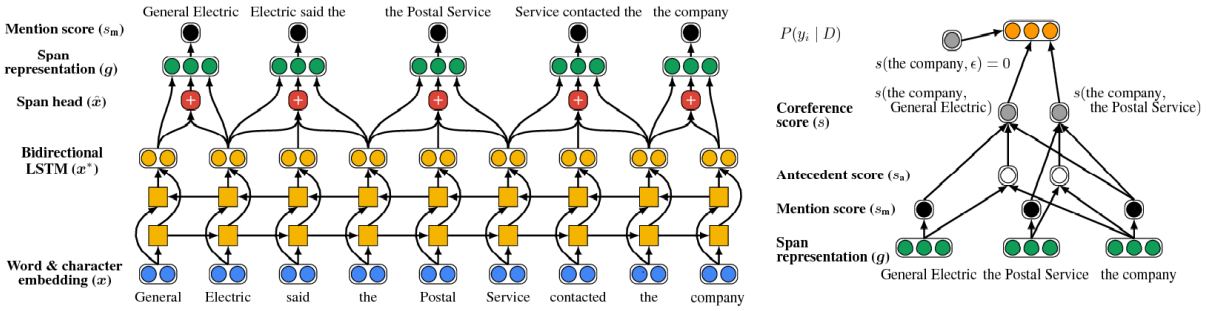

Figure 2.7: Computation of span representations and co-reference scores in the model of Lee et al. (2017).

	MUC			B^3			$CEAF_e$			CoNLL
	P	R	F1	P	R	F1	P	R	F1	Avg F1
Lee et al. (2017)	81.2	73.6	77.2	72.3	61.7	66.6	65.2	60.2	62.6	68.8
Clark and Manning (2016)	78.9	69.8	74.0	70.0	57.0	62.9	62.5	55.8	59.0	65.3

Table 2.1: Evaluation results for co-reference resolution on English CoNLL-2012 dataset.

$$\log \prod_{i=1}^{N} \sum_{y \in Y(i) \cap \text{GOLD}(i)} P(y) \tag{2.27}$$

In Equation 2.27, N is the number of possible text spans and $\text{GOLD}(i)$ is the set of spans in the gold cluster containing span i. By optimizing this objective the model prunes spans accurately, as only gold mentions receive positive updates.

Table 2.1 summarizes the results on the MUC, B^3 and $CEAF_e$ metrics, and the CoNLL score for co-reference resolution on the English CoNLL-2012 dataset, with the models presented in this section.

2.2.2 Co-Reference Resolution for Portuguese and Spanish Texts

Regarding co-reference resolution for Portuguese data, Fonseca et al. (2017a) presented CORP (Co-Reference Resolution for Portuguese), a rule based approach using lexical, syntactic and semantic knowledge. The model has a multi-step architecture, applying a rule in each step and grouping two mentions if the restrictions are satisfied. Onto.PT (Oliveira, 2012) was used to obtain semantic relations, such as hyponymy and synonymy, included in some of the rules. They start by performing mention detection using CoGrOO parser (Silva, 2013), a grammar checker which also provides syntactic annotations. Then, as a pre-processing step, they decided to remove numerical mentions (e.g. 9%, $\$100$, *dez metros*), since those are responsible for most of the wrong co-reference predictions. Finally, the following set of 13 rules (11 lexical and 2 semantic) is applied:

1. Two mentions are co-referent if their noun phrases are an exact match. This is not applica-

ble to pronouns or mentions in specifying appositive constructions that do not match (e.g. [[*o signo*][*Gemini*]], [[*o projeto*][*Gemini*]] - *Gemini* are not co-referent).

2. Two mentions are co-referent if, by truncating them considering their root, the obtained phrases are an exact match (e.g. [*o piloto americano*], [*o piloto*]). Same exceptions as the previous rule.
3. Two mentions are co-referent if they are in an explicative appositive construction. The explicative appositive is identified by parenthesis or commas (e.g. [*O Presidente da República*], [*Marcelo Rebelo de Sousa*], ...).
4. Two adjacent mentions m_i and m_{i+1} are co-referent if they are in a specifying appositive construction. To do so, they have to satisfy the following conditions: (i) m_i and m_{i+1} must be in the same sentence and be adjacent; (ii) m_i is a common noun and has a definite article; (iii) m_{i+1} is a proper noun and does not have a determiner. If the definite article of m_i is plural, it groups the following proper noun mentions, as long as they are in the same sentence and separated by commas or *e* in the last one (e.g. [*os médicos*], [*João Henriques, David Gonçalves e Sofia Silva*]).
5. Two mentions are co-referent if one is the acronym/abbreviation of the other.
6. Two mentions are co-referent if one is a nominative predicate referring to the other. To cover these cases, they search for a linking verb followed by a determiner and group the adjacent mentions (the previous is the referent and the next is the nominative predicate). The only linking verb considered was *ser* (e.g. [*Lisboa*] *é* [*a capital de Portugal*]).
7. Two adjacent mentions m_i and m_{i+1} are co-referent if m_{i+1} is a relative pronoun (e.g. [*Portugal*], [*cuja*] *população* ...).
8. Two mentions m_i and m_j are co-referent if: (i) their roots match; (ii) all the words from m_j, excluding stopwords, are included in the antecedent m_i - i.e. there are no words modifying the antecedent; (iii) m_j is not a constituent part of m_i - prepositions are used to recognize these dependencies (e.g. [*O rapaz de pijama às riscas*] and [*o pijama às riscas*] - not co-referent, *de* has to be connected to *o pijama às riscas*).
9. Two mentions m_i and m_j are co-referent if they verify the first and third conditions from the previous rule and all the modifiers (i.e. nouns and adjectives) from m_j are included in the antecedent m_i (similar to the second condition from the previous rule but less restrictive).
10. Two mentions are co-referent if they both contain proper nouns, the proper nouns match, and the mentions satisfy the third condition from rule 8.
11. Two mentions m_i and m_j are co-referent if both contain proper nouns, at least one word from m_j matches a word from m_i, and there are no words modifying the mentions (i.e. the second condition from rule 8 is satisfied).
12. Two mentions are co-referent if their root lemmas are hyponyms and there are no words modifying the mentions (similar to the previous rule).

13. Two mentions are co-referent if their root lemmas are synonyms and there are no words modifying the mentions. Additionally, each new mention to be added to an existing co-reference cluster with this rule must have a synonymy relation with all the cluster's mentions.

CORP links a mention to its antecedents if some rule is verified. Based on these co-reference pairs, clusters are formed. However, in some cases a mention can be linked to antecedents from different clusters (i.e. referring to different entities), so it is necessary to decide which cluster the mention belongs to. In this situation, CORP would erroneously output a single cluster containing all mentions.

To tackle that clustering problem, Fonseca et al. (2018) proposed a clustering method which takes into account discourse structure, using the CORP model as baseline. They assume that any mention is new in the discourse if it does not have a link to one or more antecedents. Thus, the clustering algorithm works as follows: if the mention does not have any co-reference relation, a new cluster is created; if the mention only has a co-reference relation with one cluster, it is linked to that cluster; if the mention has co-reference relations with more than one cluster, a clustering criteria is applied to decide to which one it is linked. As a clustering criteria, Fonseca et al. (2018) presented five options. Given a mention m to link, they work as follows:

- Closest Cluster: links the mention m to the closest cluster with a co-reference relation (i.e. containing the closest co-referent mention to m).
- Cluster Weight: links the mention m to the cluster with higher weight. The cluster weight is obtained by summing +1 for each CORP rule satisfied by the co-referent mentions to m from that cluster.
- Mention Weight: links the mention m to the cluster containing the greater amount of co-reference relations with it (+1 for each co-referent mention to m).
- Mention + Cluster Weight: considers the sum of the previous criteria to decide which cluster to link the mention.
- F1-Score Weight: links the mention m to the cluster with higher F1-Score weight. The F1-Score weight is obtained by summing the weight of each CORP rule satisfied by the co-referent mentions to m from that cluster. The weight of each rule corresponds to the CoNLL F1-Score obtained by applying the rule individually.

Both systems presented (CORP and CORP+Clustering) were tested on the Portuguese dataset Corref-PT (Fonseca et al., 2017b).

Regarding co-reference resolution for Spanish data, some works were developed for the SemEval-2010 Task 1 (Recasens et al., 2010). We will present some of those systems below. All the works were developed on a gold scenario regarding mention detection, using the gold mention boundaries from the dataset. The dataset used was AnCora-CO-ES (Recasens and Martí, 2010), which was already divided in training and test sets.

Kobdani and Schütze (2010) proposed SUCRE, an approach based on a relational database model and a regular feature definition language. Its architecture is divided in two parts: pre-processing and co-reference resolution. In pre-processing, the text corpus is modeled to a relational database model, which involves extracting atomic word features, detecting markables (i.e. mentions) and extracting atomic markable features. Atomic features are attributes - examples of atomic word features are the position of the word in the corpus, the document and sentence numbers, the gender and number, and the Part-of-Speech (POS) tag; examples of atomic markable features are the number of words in the markable, the syntactic role and the semantic class. Having the relational database model, co-reference resolution can be performed. Considering this, SUCRE has five functional components:

1. Relational Database Model of Text Corpus: requires at least the Word, Markable and Link tables.
2. Link Generator: for training, it generates a positive instance for each co-referent markable pair and negative instances for a markable and all its not co-referent antecedent markables.
3. Link Feature Extractor: link features are defined over a pair of markables. A regular feature definition language with some keywords is used to select different word combinations of the two markables. Some examples of the keywords used are $m1b$, $m1e$ and $m1h$, representing the first, last and head words of the first markable in the pair (similarly, $m2b$, $m2e$ and $m2h$ for the second markable). Some of the available functions for the rules are exact matching, sub-string matching, edit distance and absolute value.
4. Learning: trains a Decision Tree classifier on the train data.
5. Decoding: applicable to test data. Creates the clusters using best-first clustering - searches for the best antecedent (i.e. the one with the highest probability) predicted as co-referent.

Sapena et al. (2010) developed RelaxCor, a co-reference resolution system based on constraint satisfaction. It represents the problem as an undirected graph connecting any pair of candidate co-referent mentions. In this scenario, vertices represent the mentions and edges represent the possible co-reference between the two connected vertices. Given a pair of mentions m_i and m_j, a set of constraints restricting their compatibility is used to compute the weight of the edge e_{ij} connecting them. Each constraint has a weight associated, reflecting its confidence. The edge weight is the sum of the weights of the constraints that apply to that mention pair. The weights can be positive or negative, indicating whether the mentions are co-referent or not, respectively.

The constraints are learned automatically by evaluating a set of features over each pair of mentions in the training data. The features used are lexical (e.g. string matching of both mentions), morphological (e.g. number and gender of both mentions match), syntactic (e.g. the mentions are nested), semantic (e.g. semantic class of both mentions match), and about distance and position (e.g. distance between the mentions in sentences and mentions). The learned constraints are conjunctions of feature-value pairs, forming a positive example if the considered pair of mentions is co-referent, and a negative one

otherwise. The weight associated with each constraint is the fraction of co-referent examples where the constraint applies minus a balance value (0.5).

RelaxCor uses relaxation labeling over the set of constraints for the resolution process. Relaxation labeling is an iterative algorithm that performs function optimization based on local information. It maintains a vector with a probability distribution for each mention, where each value corresponds to the probability of the mention belonging to a specific partition (i.e. entity) given all the possible partitions. During the resolution process, these probability vectors are updated taking into account the edge weight and the probability vectors of the adjacent mentions. The larger the edge weight, the greater the influence of the adjacent probability vector. The algorithm updates the probability vectors in each step until convergence (i.e. no more significant changes are done). The final partition is directly obtained by assigning each mention to the partition with the highest probability.

Attardi et al. (2010) proposed TANL-1, a co-reference resolution system using a binary classifier and a greedy clustering technique. A Maximum Entropy classifier is trained to determine whether two mentions refer to the same entity or not. Regarding the training instances, a positive instance is created by pairing each mention with each of its co-referent antecedents, and a negative instance is created by pairing each mention with each of its preceding non co-referent mentions. The classifier is trained using the following features, for each pair of mentions:

- Lexical: whether the two mentions are equal, one is a prefix of the other, one is a suffix of the other, one is the acronym of the other, and the edit distance between them.
- Distance: sentence, token and mention distance between the mentions.
- Syntax: whether the heads of the two mentions have the same POS, and a pair with the POS of the two mentions heads.
- Count: a pair of numbers, each counting how many times a mention occurred.
- Type: whether the two mentions have the same Named Entity (NE) type.
- Pronoun (if the most recent mention is a pronoun): a pair with the two mentions' gender attributes, a pair with the two mentions' number attributes, and the pronoun type.

According to the output of the classifier, the mentions are clustered using best-first clustering.

All the previously presented models were developed on a monolingual scenario and using rule based or ML approaches. Not many co-reference resolution neural models have been developed for Portuguese or Spanish, nor models exploring cross-lingual learning for these languages. However, recently, Cruz et al. (2018) developed state-of-the-art systems for co-reference resolution on Spanish and Portuguese data, exploring a cross-lingual setting: direct transfer learning from Spanish to Portuguese.

Their work was focused on the classification phase of co-reference resolution, so the model was supplied with gold standard mention boundaries, not dealing with mention detection. The linking algorithm used was *closest antecedent*, which links each mention to its closest positively identified antecedent, if

there is one.

They subdivided the proposed neural models in two steps: (i) extracting representative features for mentions, which is performed using Convolutional Neural Networks (CNNs), LSTMs or dense layers; (ii) assessing co-reference affinity, which is performed using dense layers. In the first step, the models receive as input a pair of 50-dimensional vectors, imposing a limit of 50 words per mention. Mentions with more than 50 words are cropped (a very small percentage) and mentions with less than 50 words are padded with a special embedding filled with zeros. The distance in sentences and tokens between both mentions is also passed as an additional feature in the input. Given this, they tested five different model variations:

1. Arch1: composed by an embedding layer whose vectors were obtained from a pre-parsed FastText file, containing the most common words at training; word embeddings are summed getting a mention embedding; the embeddings from both mentions are stacked and passed through a standard 1D convolutional layer; the obtained representation is concatenated with scalar distance features and passed through two fully-connected layers: a standard one and a final sigmoid-activated one, which is the output layer.
2. Arch2: the embedding layer is created by tokenizing all the texts from the input dataset, loading the entire embeddings model and using FastText ability to predict embeddings for the out-of-vocabulary words; remaining layers are still the same as Arch1.
3. Arch2-dense: the embedding layer is the same as Arch2; resulting embeddings are also summed and then concatenated with the distance features; the obtained vector is passed through two hidden layers and the final output layer.
4. Arch-deep-CNN: the embedding layer is the same as Arch2; to obtain the mention representation, instead of summing the word embeddings, their vectors are passed through two 1D convolutional layers; the obtained vectors are max-pooled along the first axis, and then passed through two hidden layers and the final output layer.
5. Arch-BiLSTM: the embedding layer is the same as Arch2; then, the word embeddings are fed into a BiLSTM layer; the last state of the LSTM is extracted and passed through two hidden layers and the final output layer.

All hidden and convolutional layers are activated by a ReLU funtion. For the word embeddings, they used pre-trained FastText multilingual word vectors (Grave et al., 2018), whose vector spaces were aligned after training, meaning the Portuguese and Spanish versions of a word have close vector representations in their respective embedding spaces.

To obtain the training instances, Cruz et al. (2018) created pair-wise combinations of mentions by pairing each mention with all its candidate antecedents. An instance is created for every pair of mention and antecedent, adding a third element specifying their co-reference relation: $(mention, antecedent, P)$

		MUC			B^3			$CEAF_e$			CoNLL
		P	R	F1	P	R	F1	P	R	F1	Avg F1
ES	Rand1	10.7	8.2	9.3	63.7	52.1	57.3	45.5	55.8	50.1	38.9
	Rand2	2.5	4.7	3.2	62.5	80.9	70.5	73.5	57.0	64.2	46.0
	AlwaysNo	0	0	0	62.2	**100**	76.7	**89.2**	55.5	68.4	48.4
	TANL-1	56.5	16.6	25.7	93.4	65.2	76.8	64.7	66.9	65.8	56.1
	RelaxCor	**73.8**	14.8	24.7	**97.5**	65.3	78.2	66.6	66.6	66.6	56.5
	SUCRE	58.3	52.7	**55.3**	79.0	75.8	77.4	69.8	69.8	69.8	67.5
	Arch-BiLSTM	42.7	**65.7**	51.6	72.2	86.6	**78.8**	87.5	**72.3**	**79.2**	**69.9**
PT	Rand1	13.7	20.5	16.5	30.5	54.2	36.0	46.2	24.1	31.7	29.1
	Rand2	3.6	11.7	5.4	27.2	79.5	40.6	51.7	17.6	26.2	24.1
	AlwaysNo	0	0	0	26.4	**100**	41.7	52.2	13.8	21.8	21.2
	CORP	44.2	52.2	47.9	35.8	45.8	40.2	46.1	43.9	44.9	44.3
	CORP+Clustering	**54.9**	50.2	**52.5**	**51.8**	43.6	47.3	46.2	**52.8**	**49.3**	**49.7**
	Arch2	46.8	**59.7**	**52.5**	47.0	62.6	**53.7**	**55.1**	34.5	42.4	49.5
Direct Transfer (ES-PT)	Arch2	**56.9**	**60.9**	**58.7**	**58.6**	**39.7**	**45.8**	**33.0**	**28.0**	**29.7**	**44.8**

Table 2.2: Evaluation results for co-reference resolution on AnCora-CO-ES and Corref-PT datasets.

if positively co-referent, or $(mention, antecedent, N)$ if not. Since this generates a highly unbalanced dataset, with more non co-referent instances, they used a random undersampling of that class. The undersampling percentage which was able to maximize model the performance was 70%.

As baselines for evaluation, they developed two random approaches and one deterministic. In the two random baselines (Rand1 and Rand2), for each mention, a random antecedent mention is chosen with x probability, otherwise no antecedent is selected for that mention. For the first random approach, Rand1, $x = 50\%$, while for the second one, Rand2, x is the percentage of co-referent mentions in the corpus. In the deterministic approach, AlwaysNo, they set all mentions as having no antecedent (i.e. singleton mentions).

The two datasets used were AnCora-CO-ES for Spanish, and Corref-PT for Portuguese. As mentioned before, the AnCora-CO-ES corpus was already split, so the results were reported on the test set, the development set was used for validation and the training set was used to train the models. On the other hand, the Corref-PT corpus was not split, so they randomly selected 60% for training, 20% for development and 20% for testing.

Table 2.2 summarizes the results on the MUC, B^3 and $CEAF_e$ metrics, and the CoNLL score for the systems presented in this section. For CORP+Clustering (Fonseca et al., 2018), we report the results using the Cluster Weight criteria, which presented the best results. Similarly, for the work of Cruz et al. (2018) we only report results for the baselines and the architecture with the best performance. Additionally, exploring a cross-lingual scenario, Cruz et al. (2018) experimented direct transfer of model weights from Spanish to Portuguese by training the model on the Spanish data and testing it on the Portuguese test set. The results of the architecture with the best performance for direct transfer learning are also reported in Table 2.2. All these results will be used for comparison with the proposed model.

3

Co-Reference Resolution for Portuguese and Spanish Texts

Contents

This chapter details the proposed model for co-reference resolution in Portuguese and Spanish texts. Section 3.1 describes the architecture of the model and the methods used for mention detection, post-processing of word embeddings and data augmentation. Section 3.2 presents the cross-lingual word embeddings used. Finally, Section 3.3 explains the hyperparameters and training strategy adopted for the model.

3.1 The Proposed Model

In this section we will present the developed model for co-reference resolution in Portuguese and Spanish texts, which is an adaptation of a previous one: NeuralCoref[1]. Figure 3.1 shows the overview of our system.

NeuralCoref is an extension for SpaCy, implementing a state-of-the-art neural co-reference resolution system. It uses NNs to resolve co-reference clusters and includes simple contextual information in mention representations. NeuralCoref is an adaptation of Clark and Manning (2016) work, presented in Subsection 2.2.1, more specifically of its mention-pair encoder and mention-ranking model.

Its co-reference resolution algorithm is divided in three steps: (i) extracting a series of mentions (pronouns, noun phrases and named entities); (ii) computing a set of features for each mention and pair of mentions; (iii) finding the most likely antecedent for each mention (if there is one) based on the set of features. The second and third steps are the ones adapted from Clark and Manning (2016) model.

The first step is performed by a rule based mention detection function, using SpaCy Tagger, Parser and Named Entity Recognition (NER) annotations to identify the potential mentions. The mention detection rules defined in NeuralCoref were specific for English texts, so for our model we modified this function. We will explain it in detail in in Subsection 3.1.1.

Once all the potential mentions are identified, the model reaches the second step: extracting a set of features for each mention and each pair of mentions. This is performed using word embeddings and some additional integer and boolean features. To include some simple contextual information about the mentions in the features, the model takes embeddings for several words inside and around each mention and averages them, generating span vectors.

The single mention features are:

- The type of the mention (e.g. noun, etc.);
- If the mention is nested;
- The type of the document (e.g. news, etc.);
- The location of the mention;
- The size of the mention;

[1] https://github.com/huggingface/neuralcoref

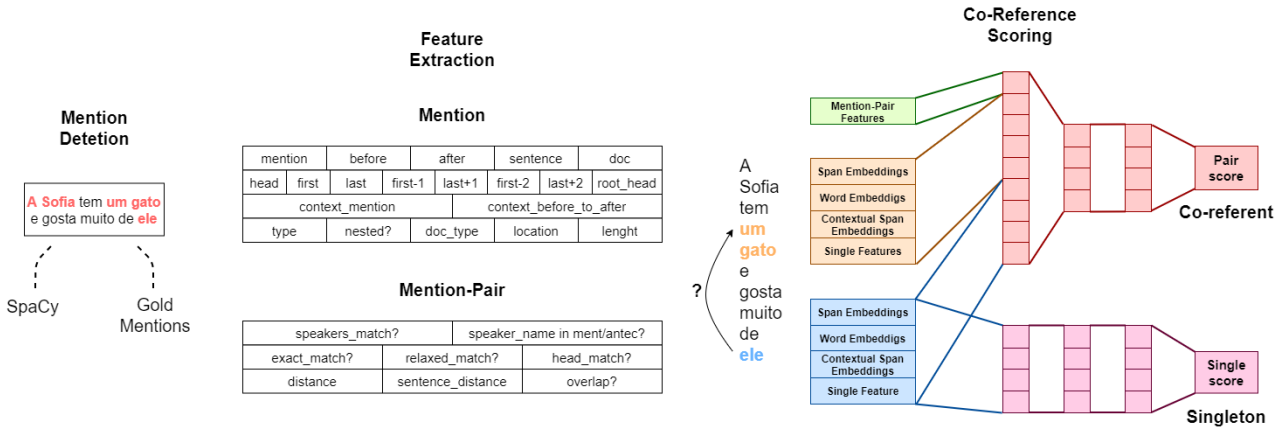

Figure 3.1: Overview of the co-reference resolution system proposed.

- Indices for the word embeddings of the following mention elements: root, first word, last word, previous word, next word, second previous word, second next word, root head;
- Span vectors for the following elements: mention, five words before the mention, five words after the mention, sentence, document.

For a mention pair, the features are:

- If the mentions have the same speaker;
- If the mention speaker's name is in the antecedent;
- If the antecedent speaker's name is in the mention;
- If there is an exact string match between them;
- If there is a relaxed string match between them (i.e. nouns/proper nouns match);
- If there is a match between the mentions' roots;
- The distance between them;
- The sentence distance between them;
- If the mentions overlap.

Finally, these features are concatenated and fed into two NNs, entering the third step. The model has a common embedding layer that transforms the words embedding indices (one of the mention features explained above) in word vectors, before feeding the NNs. The first NN computes a score for each pair of a mention and a possible antecedent, taking as input the single mention features for each mention, along with their pair features. The second NN computes a score for a mention having no antecedent, taking as input its single mention features. All these scores are compared and the highest determines if the mention has an antecedent and if so, which one.

The training goes through three successive phases: All-Pairs, Top-Pairs and Ranking. The model moves on to the next stage when the established number of epochs is over or if the evaluation metric

on the development set stops increasing for three epochs. When changing to the next stage, the best model from the previous stage is loaded. The first phase, All-Pairs, uses a cross-entropy loss on the full set of mention pairs. The second phase, Top-Pairs, also uses a cross-entropy loss but only on the top scoring antecedents (true and false) of a mention. The third and last phase, Ranking, uses a max-margin loss with slack-rescaled costs. These learning phases are similar to the ones used in Clark and Manning (2016) mention-ranking model, so their loss functions can be consulted in Equations 2.20, 2.21 and 2.18 respectively.

NeuralCoref was developed to process English data (CoNLL-2012 corpus), so for our model, in addition to changing the rules for mention detection, we adapted it to read the Portuguese and Spanish corpora and to work in a cross-lingual scenario, using it as base model.

3.1.1 Mention Detection

For the mention detection task, we relied on SpaCy models for each language, providing POS Tagger, Parser and NER annotations. This information is used on a set of rules to identify candidate mentions. For Portuguese we used the *pt_core_news_sm* model and for Spanish we used the *es_core_news_sm* model. We started by parsing the corpus documents with SpaCy. From the SpaCy parsed doc, we were able to extract spans (i.e. phrases), which we analyzed one by one to obtain the candidate mentions - pronouns, noun phrases and named entities.

For each span, we went through each token applying the following set of rules:

1. Verify if the coarse-grained POS tag corresponds to a noun (NOUN), a proper noun (PROPN) or a pronoun (PRON), or if the syntactic dependency label corresponds to a nominal subject, an indirect object or an object. If this does not verify, move on to the next token.
2. If the token is a personal or relative pronoun, add it a candidate mention.
    i. If the pronoun is part of a conjunction, obtain the span that goes from its leftmost to its rightmost syntactic descendants and add it as a candidate mention.
    ii. Move on to the next token.
3. Obtain the leftmost and rightmost token's syntactic descendants. Take the span that goes from the left one to the right one and clean it, by verifying if it does not start or end in punctuation, conjunctions, interjections or prepositions. If the start (or end) is not valid, move it to the next (or previous, respectively) position, until reaching a valid span or an empty one. If a valid span is obtained, add it as a candidate mention. Otherwise, move on to the next token.
4. Given the previously obtained span, verify if there is punctuation in it. If so, separate a segment containing the token being processed, whose limits correspond to the closest punctuation (on each side), not including them (e.g. previous span: *o presidente, Marcelo Rebelo de Sousa*, initial

token: *presidente*, separated segment: *o presidente*). Clean the separated span (as explained above) and, if it is valid, add it as a candidate mention. Otherwise, move on to the next token.

5. Given the previously obtained span, verify if there are conjunctions or prepositions in it. If so, separate a segment containing the initial token, whose limits correspond to those terms (the closest ones), not including them (e.g. previous span: *o gato e o cão*, initial token: *cão*, separated segment: *o cão*). Clean the separated span (as explained above) and, if it is valid, add it as a candidate mention. Otherwise, move on to the next token.
6. Finally, if the span obtained from the previous steps starts and/or ends with a verb, remove the verb (e.g. previous span: *a equipa comandada*, new span: *a equipa*). Clean the obtained span and, if it is valid, add it as a candidate mention.

We started by implementing rules 1, 2, 3 and 5 as a base. Comparing the obtained candidate mentions to the gold ones, we observed that a considerable number of candidate mentions contained punctuation in them and some smaller gold mentions within those were not recognized as candidate mentions. We covered those cases by adding rule 4 to the previous ones. We also noticed that some examples like the one described in rule 6 were happening, in which some candidate noun phrases included a verb classifying the noun, but the gold mentions did not, so we added that rule, increasing the number of correct mentions detected on both languages.

In addition to these rules, SpaCy is able to recognize NEs. For Portuguese and Spanish, the accepted entities can represent a named person or family (PER), a name of politically or geographically defined location (LOC), a named corporate, governmental, or other organizational entity (ORG), and miscellaneous entities - events, nationalities, etc, (MISC). In both languages, we added those NEs as candidate mentions. Analyzing some examples from each language dataset separately, we concluded that some additional specific rules could be added.

For Spanish, we noticed that the used dataset contained underscores as mentions to missing subjects (e.g. *Lo peor que* [_] *podemos hacer a un menor de edad*). Most of the times, those underscores were identified as proper nouns or syntactically classified as nominal subjects or objects, as they were supposed to. However, going through the rules, sometimes the candidate mentions recognized did not include the underscores individually, only spans with them surrounded by other words. To overcome this, we added a rule to verify if the token being processed was an underscore, and if so we added it as a candidate mention.

For Portuguese, we observed that our function was considering parts of a proper noun as candidate mentions, along with the complete proper noun (e.g. candidate mentions: [[*Maria*] [*Joana*]], gold mention: [*Maria Joana*]). Since the dataset does not consider these cases as mentions, we added a rule verifying if a candidate mention corresponding to a proper noun was contained in another one, also corresponding to a proper noun. If so, the contained mention was removed from the candidate

mentions, reducing the number of wrong candidates generated. We also noticed that in examples like *a atleta Maria* (a nominal modifier followed by a proper noun) our rules detected [*a atleta* [*Maria*]] as candidate mentions, instead of [*a atleta*] [*Maria*] as in most of the gold annotations. To fix this, we added a rule verifying if a candidate mention corresponding to a proper noun was contained in another candidate mention. If so, we took the part differentiating them (i.e. *a atleta*) and cleaned it. If the obtained span was valid, we added it as a candidate mention. Despite this generating several more candidate mentions, it also adds more correct candidates, maintaining the ratio between them.

This complete set of rules can generate duplicates, so by the end we cleaned the candidate mentions to eliminate them.

3.1.2 PCA Projection

Following the work of Mu et al. (2017), we decided to do a post-processing for the word embeddings, in which the objective was to obtain more discriminative representations. The post-processing consists in eliminating from the word vectors: the common mean vector and a few top dominating directions, obtained through Principal Component Analysis (PCA). The idea behind it is that all word representations share a same common mean vector and have the same dominating directions. Such vector and directions strongly influence the word vectors in the same way, so by eliminating them the representations capture more discriminative information.

Given the word representations $\{r(w), w \in V\}$, V being the vocabulary and w a word, the post-processing algorithm goes as follows:

1. Compute the mean of the word representations: $\mu = \frac{\sum_{w \in V} r(w)}{|V|}$
2. Update the word representations by subtracting the mean from them: $\tilde{r}(w) = r(w) - \mu$
3. Using the updated word representations, compute D PCA components (the dominating D directions): $c_1, ..., c_D = \mathrm{PCA}(\{\tilde{r}(w), w \in V\})$
4. Process the representations eliminating the D dominant directions: $r'(w) = \tilde{r}(w) - \sum_{i=1}^{D} (c_i^{\mathsf{T}} r(w)) c_i$

Regarding the dominant directions, D depends on the representations (e.g. their dimension, the training methods) and on the downstream application. Mu et al. (2017) suggest that choosing D around $d/100$ for word representations with dimension d works well across multiple languages, representations and applications. Since our representations have a dimension $d = 300$, we started testing with $D = 3$. After that, we tried to increase the value to $D = 4$ and $D = 10$, obtaining worse results overall. Finally, we tested $D = 2$, but the results were also lower. Considering this, for our model we selected $D = 3$.

3.1.3 Cross-Lingual Contextual Embeddings

As previously explained, NeuralCoref only includes some simple contextual information about the mention in the features by including the embeddings of some words around it. For our model, we created an additional single mention feature, whose objective was to provide more contextual information through contextual embeddings. For that purpose, we used sentence-transformers library[2], which provides models based on Transformer networks capable of computing contextual sentence representations. Since we were working with two languages, we focused on the multilingual models, namely: *distiluse-base-multilingual-cased*, a multilingual knowledge distilled version of Multilingual Universal Sentence Encoder (mUSE).

Multilingual knowledge distillation (Reimers and Gurevych, 2020) is a method that allows creating multilingual versions from previously monolingual models. It can also be applied to previous multilingual models in order to expand the number of supported languages. Its training is based on the idea that a translated sentence should be mapped to the same location in the vector space as the original sentence (meaning the vector spaces are aligned). The method requires a fixed teacher model M_t, that produces sentence embeddings with the desired properties in one or more source languages. It also requires a set of parallel translated sentences $((s_1, t_1), ..., (s_n, t_n))$, with s corresponding to sentences in one of the source languages and t to sentences in one of the target languages. With these resources a multilingual student model M_s is trained to mimic the teacher one, such that the same sentence is mapped to the same vector by both models: $M_t(s_i) \approx M_s(s_i)$ and $M_t(t_i) \approx M_s(t_i)$. Mean-Squared Error (MSE) is used to train the model.

For *distiluse-base-multilingual-cased*, the teacher model used was mUSE Yang et al. (2019) and the multilingual student model used was XLM-RoBERTa, pre-trained on 100 languages. mUSE computes multilingual sentence embeddings using a dual-encoder Transformer architecture. The Transformer encoder is used to compute context-aware representations of the tokens in a sentence, which are then averaged together to obtain the sentence embedding. It was trained for 16 languages in a multi-task setup, including question-answer prediction, translation ranking and natural language inference.

XLM-RoBERTa was trained to mimic Multilingual Universal Sentence Encoder (mUSE) with parallel data for 50 languages (including Portuguese and Spanish), obtaining the distilled mUSE model.

We used *distiluse-base-multilingual-cased* to compute span embeddings for the new feature. Based on the existent feature containing span vectors, we tried the following variations:

1. Embedding for the mention.
2. Embedding for the span going from the fifth word before the mention to the fifth word after.
3. Embeddings for: the mention, the span going from the fifth word before the mention to the fifth word after.

[2]https://github.com/UKPLab/sentence-transformers

4. Embeddings for: the mention, the five words before the mention, the five words after the mention, the sentence and the document.

The option presenting better results was the third new feature variation, so we added it to our model.

3.1.4 Data Augmentation

As we explained before, one of the challenges for Portuguese co-reference resolution is the shorter amount of annotated data available. To explore a solution for that problem we decided to do data augmentation, translating the Spanish training data to Portuguese and vice-versa, getting more training data for each language. For the translator we used the googletrans library[3]. Algorithm 1 shows our approach for translating each sentence in the documents.

Algorithm 1 Translation per sentence

1: Translate sentence s - t_s
2: **for each** mention m in s **do**
3: Translate m - t_m
4: **if** there are matches of t_m in t_s **then**
5: Count the matches - $count$
6: Get the match whose $t_m's$ 1^{st} word position in t_s is closer to $m's$ 1^{st} word position in s and annotate the distance ($dist$)
7: **if** $count = 1$ **and** (m only appears once in s **or** $dist \leq 3$) **then**
8: Annotate the match in t_s
9: **else if** $count > 1$ **and** $dist \leq 3$ **then**
10: Annotate the match in t_s
11: **else**
12: **for** fuzzy_match of t_m in t_s **do**
13: Do steps 5-6
14: **if** $count > 0$ **and** $dist \leq 3$ **then**
15: Annotate the match in t_s

The main idea behind it is that Portuguese and Spanish are similar languages with similar sentence constructions. So, when translating a sentence, the translation should have a similar structure, with the corresponding words in close positions. Step 6 of the algorithm is based on that notion: the match the with closest position is the most likely to be correct, so it is chosen. Following this idea, we defined a threshold of 3 for the position difference. We tested higher values, but the greater the margin, the greater the number of wrong annotations. This threshold is a balance between flexibility and wrong annotations. We made an exception for one-to-one matches (i.e. only one match in the translation and the original sentence), as shown in step 7. We assumed those were right, regardless of their positions. When annotating a match, it is possible that there is already another annotation with the same boundaries. In those cases, only one can be correct, so we compared their corresponding $dist$ values and picked the annotation with the smallest.

[3]https://pypi.org/project/googletrans/

We tried using only exact matches, but those are a small percentage (less than 50%). One of the reasons behind it is that mentions are translated without their context, generating small differences when compared to its correspondent in the sentence. To allow those small differences, we identified fuzzy_matches using fuzzywuzzy[4] and fuzzysearch[5] libraries.

We used fuzzywuzzy to verify if the fuzzy match between t_m and t_s was above a score threshold. Fuzzywuzzy measures sequence similarity on a scale from 0 to 100, weighting different algorithms (e.g. SequenceMatcher Ratio, PartialRatio, SortedRatio) and selecting the best score. Since we only wanted very similar cases, we tried setting the threshold to 90. However, very few matches were above it so we lowered it to 80, accepting more matches, the vast majority of them correct.

If the fuzzy match between t_m and t_s was above the threshold, we used fuzzysearch to find the subsequences in t_s that approximately match t_m. Fuzzysearch uses Levenshtein Distance for that purpose, imposing a threshold for a maximum distance. We set that maximum value to 5, allowing only small changes or typos (e.g. a missing preposition or article).

The data obtained from Spanish translation to Portuguese ended up with 41000 mentions (52% of the original annotations) and the data from Portuguese translation to Spanish with 5314 mentions (59%). This represents a good increase in the amount of data for Portuguese. Although we used more restrictive conditions, the translation algorithm still allows some mention annotation errors, as expected.

3.2 Cross-Lingual Word Embeddings

For our model, we focused on developing a cross-lingual system for Spanish and Portuguese, so we used the MUSE library to obtain the cross-lingual word embeddings.

The library already provides embeddings for Portuguese and Spanish aligned on a single vector space; however, they only make available a text file with the embeddings of the words they used in the alignment. Since it is not possible to obtain the embeddings for out-of-vocabulary words and not all words in our training data are present there, we redid the alignment. As there is more Spanish data available, it was used as the source language and Portuguese was used as the target language for the alignment. We used the Supervised method available in the MUSE library, which takes a bilingual dictionary and learns a mapping from the source to the target space using (iterative) Procrustes alignment. The library already had available a bilingual dictionary for Portuguese and Spanish (obtained using the Unsupervised method), which we used to align our embeddings. We chose the Supervised method over the Unsupervised one since all the necessary resources were available and the former is faster (skips the adversarial learning step).

[4] https://pypi.org/project/fuzzywuzzy/
[5] https://pypi.org/project/fuzzysearch/

In addition to the bilingual dictionary, the Supervised method also takes as input text files with embeddings for each language. Those files must contain a very considerable amount of samples (i.e. word embeddings), so we can obtain a good alignment. To obtain those input files, we started by getting pre-trained FastText word vectors for each language[6]. The word vectors were available in two formats: a text file and a binary model, which can be used to obtain vectors for out-of-vocabulary words. We gathered all the FastText word embeddings in the text file, and added the ones corresponding to the out-of-vocabulary words (present in the training data and not in the file), loading the binary model and using FastText ability to predict their embeddings. We did this for both Portuguese and Spanish, using the obtained files as input for the MUSE model. The model outputs a text file with the Portuguese MUSE embeddings and another one with the Spanish MUSE embeddings, aligned in the same vector space. The embeddings in each output file represent the words given in the corresponding input file.

Since we used a lot of words for the alignment, in the end we filtered each output file obtained, including just the words in the respective training data, using those versions in our model.

3.3 HyperParameter Choices and Model Training Strategy

Regarding the model hyperparameters, we kept the slack-rescaled costs used in the NeuralCoref ranking loss: 0.8 for a false new, 0.4 for a false link and 1 for a wrong link. We set the initial learning rate for each training phase to 10^{-3}, the minimum learning rate to 10^{-8} and the patience to 5. We kept these values fixed during our training, not tuning them.

We also made some changes in the NeuralCoref training strategy. During our training we used the CoNLL F1-Score on the development set as evaluation metric. For a training strategy analysis, we used the base version of our model without the additional methods (PCA, Distilled mUSE and Data Augmentation), supplied with gold mention boundaries.

We started by changing the criteria to move between the three training phases. We added a call-out function which lowers the learning rate by 10^{-1} if the CoNLL F1-Score has no improvements for more than 5 epochs (patience), until the learning rate reaches the minimum value. Only then, after more than 5 epochs without improving the CoNLL F1-Score with the minimum learning rate, we would go to the next training phase. We did not specify a maximum number of epochs for training or any early stop condition. We used Adam (Kingma and Ba, 2017) as the optimizer for training.

We trained the model with these parameters and criteria, and we noticed that the second phase (Top-pairs) was lowering the development set CoNLL F1-Score instead of improving it, comparing to the results from the first phase (All-Pairs). Given this information, we tried to eliminate the second training step, working only with the other two, which presented improvements on the CoNLL F1-Score.

[6] https://fasttext.cc/docs/en/crawl-vectors.html

Analyzing in more detail the epochs, we observed that the best CoNLL F1-Score for each training phase was obtained in their first few epochs, without improving on the following ones.

To try to improve the results over the epochs, we implemented a mixed training, switching between All-Pairs and Ranking phases - train 5 epochs with All-Pairs loss, then 5 epochs with Ranking loss, 5 epochs with All-Pairs loss again, and so on - loading the best model at switching and maintaining the learning rate updates as before. This showed mild improvements, again obtained on the early epochs. Finally, we changed the mixed training strategy to switch between All-Pairs and Ranking when the CoNLL F1-Score stops improving, instead of running a specific number of epochs before that. The model trained with this strategy achieved the best results so we used it as the Base model of our proposal.

4

Experimental Evaluation

Contents

This chapter describes the experimental evaluation of the proposed model. Section 4.1 presents the evaluation methodology. Section 4.2 introduces the datasets used in the experiments. Section 4.3 details the evaluation metrics used to evaluate the model performance. Finally, Section 4.4 contains the obtained results and a discussion on them.

4.1 Methodology

To focus on evaluating our model on the classification task of co-reference resolution, we ran experiments supplying the model with gold mention boundaries covering the two scenarios: monolingual and cross-lingual training.

For the monolingual scenario we trained the model on Portuguese and Spanish data separately. For Portuguese we used the Portuguese MUSE embeddings and for Spanish we used the Spanish MUSE embeddings. We tested adding each proposed method to the Base model individually and then combining them. The results are reported on the test portion of each dataset.

For the cross-lingual scenario we trained the model simultaneously with Portuguese and Spanish data. We used both Portuguese and Spanish MUSE embeddings. For words appearing in both files we used the Spanish representation (either one should work since they are aligned). Similarly, we tested adding each proposed method to the Base model individually and then combining them. We reported the results on Portuguese and Spanish test sets individually, to evaluate the performance of the cross-lingual model on each language.

We also ran experiments in both the monolingual and cross-lingual scenarios with our mention detection mechanism instead of gold mention boundaries. The mention detection algorithm generates many candidates, requiring more computational resources, such as time and memory. Since the additional proposed methods further increase the resources needed (e.g. adding the contextual embeddings feature implies bigger feature vectors and data augmentation implies more mentions), we only tested this variant on the Base model.

4.2 Datasets

Similar to the work of Cruz et al. (2018), the datasets used to train and test our model were Corref-PT for Portuguese, and AnCora-CO-ES for Spanish. Corref-PT is the largest Portuguese dataset annotated containing co-reference information (Brazilian variant, since European corpora are rarer). Similarly, AnCora-CO-ES is the largest dataset available for Spanish containing co-reference information. Table 4.1 shows the number of tokens and documents of each dataset.

We also did a similar pre-processing of both datasets. The Spanish corpus was already divided, so

Dataset	Language	#Tokens	#Docs
AnCora-CO-ES	Spanish	380K	1183
Corref-PT	Portuguese	124K	182

Table 4.1: Datasets used in the evaluation experiments.

we used the corresponding train, development and test sets. For the Portuguese corpus, we divided its documents randomly, using 60%, 20% and 20% for the train, development and test sets, respectively.

Additionally, we analyzed some examples in each dataset. We noticed that some names in both datasets were written in a single line with underscores separating their words, instead of having each separate word in a different line (e.g. *Instituto_de_Agronomia*, instead of *Instituto* / *de* / *Agronomia*). This affects SpaCy's annotations for those tokens, making it harder to correctly detect those mentions. Additionally, the embeddings for both are different, since the first would use just one word embedding and the second would use the average for the three word embeddings. Having the separate embeddings can help the model detecting similarity with a co-referent mention using one of the words (e.g. a next reference as *o Instituto*). For these reasons, we decided to modify both datasets and separate the words in the underscores without affecting the co-reference annotations, by starting them on the first word and ending on the last one.

We also observed that some annotated gold mentions started and/or ended with a punctuation mark. The mention detection function cleans the candidates, in order for them not to start/end in punctuation (e.g. *a casa* instead of *a casa ,*), so those examples would never be considered as candidate mentions. Since the punctuation does not change the mention, we considered those cases simple errors, so we corrected them in both datasets by changing the mention start (or end) to the next (or previous, respectively) word.

4.3 Evaluation Metrics

To evaluate our model, we reported results on the three most common metrics used for the evaluation of co-reference resolution systems: MUC, B^3 and $CEAF_e$. We also reported the results on the CoNLL metric, which combines the previous ones. These were the metrics reported in the related works presented, allowing us to make comparisons.

MUC, B^3 and $CEAF_e$ are defined in terms of how they calculate Precision and Recall. For the three metrics, the F1-Score is computed as a harmonic mean of Precision and Recall. The CoNLL metric is the mean of the three F1-Scores.

Consider $K = k_i : i = 1, 2, ..., |K|$ as the gold entity set and $S = s_i : i = 1, 2, ..., |S|$ as the predicted entity set. k_i and s_i represent the entities (i.e. clusters), with $|K|$ and $|S|$ being the number of mentions.

MUC (Vilain et al., 1995) is a link based metric that operates by comparing the entities defined by the

gold links and the predicted links, instead of the links themselves. Recall (or Precision) is based on the minimum number of links that need to be added to the predicted entities (or gold entities, respectively), in order to get them aligned with the gold ones (or predicted ones, respectively). The following equations show how to compute Precision and Recall:

$$\text{Precision} = \frac{\sum_{s_i \in S}(|s_i| - |p(s_i)|)}{\sum_{s_i \in S}(|s_i| - 1)} \tag{4.1}$$

$$\text{Recall} = \frac{\sum_{k_i \in K}(|k_i| - |p(k_i)|)}{\sum_{k_i \in K}(|k_i| - 1)} \tag{4.2}$$

In Equation 4.2, $p(k_i)$ is a partition of k_i containing subsets of it, where each subset is created by intersecting k_i with the predicted entities that overlap with it. Similarly, in Equation 4.1, $p(s_i)$ is a partition of s_i relative to the gold standard.

B^3 (Bagga and Baldwin, 1998) measures performance on the mention level. It computes individual Precision and Recall for each mention and then computes the average of these values to get the final Precision and Recall, as shown in the following equations:

$$\begin{aligned} \text{Precision}(m_i) &= \frac{|k_{m_i} \cap s_{m_i}|}{|s_{m_i}|} \\ \text{Precision} &= \frac{\sum_i^{M_s} \text{Precision}(m_i)}{M_s} \end{aligned} \tag{4.3}$$

$$\begin{aligned} \text{Recall}(m_i) &= \frac{|k_{m_i} \cap s_{m_i}|}{|k_{m_i}|} \\ \text{Recall} &= \frac{\sum_i^{M_k} \text{Recall}(m_i)}{M_k} \end{aligned} \tag{4.4}$$

For each mention m_i, Recall (or Precision) computes the number of correct mentions in the predicted entity containing m_i over the number of mentions in the gold entity (or predicted entity, respectively) containing m_i. In Equations 4.3 and 4.4, s_{m_i} and k_{m_i} represent the predicted and gold entities containing m_i, respectively. When computing the final measures, M_s and M_k are the numbers of predicted and gold mentions, respectively. Using gold mention boundaries, $M_s = M_k$.

However, when using a mention detection system, there are *twinless mentions* - predicted mentions that are not mapped to any gold mention and vice-versa. To overcome that problem, when performing mention detection the following modifications were incorporated in the scorer (Cai and Strube, 2010):

- Include the non-detected gold mentions in the prediction as singletons.
- Discard the detected mentions not included in the gold ones and resolved as singletons.
- Computing Recall: discard the *twinless predicted mentions* in the predicted mentions set.
- Computing Precision: add the *twinless predicted mentions* to the gold mentions set as singletons.

CEAF (Luo, 2005) computes the alignment between gold and predicted entities. It finds the best one-to-one mapping between gold and predicted entities, using a similarity measure (ϕ) for each pair of entities to determine the value of each possible alignment. Every predicted entity is aligned with at most one gold entity. The best mapping function g^* (i.e. the one with the highest total similarity) is used to compute Precision and Recall, as shown in Equations 4.5 and 4.6. K^* is the set of gold entities included in the best mapping.

$$\text{Precision} = \frac{\sum_{k_i \in K^*} \phi(k_i, g^*(k_i))}{\sum_{s_i \in S} \phi(s_i, s_i)} \tag{4.5}$$

$$\text{Recall} = \frac{\sum_{k_i \in K^*} \phi(k_i, g^*(k_i))}{\sum_{k_i \in K} \phi(k_i, k_i)} \tag{4.6}$$

We used the entity based variant, CEAF_e, which computes the similarity as the relative number of common mentions between the two entities: $\phi(k, s) = \frac{2 \cdot |k \cap s|}{|k| + |s|}$. With CEAF_e, the denominator of Equations 4.5 and 4.6 corresponds to the number of predicted and gold entities, respectively.

4.4 Results

Table 4.2 reports the results of the proposed model trained on a monolingual scenario for each language, using gold mention boundaries.

Regarding the monolingual training on Spanish data, the models combining Base+DmUSE and Base+PCA+DmUSE+DA had the best performances, with 74.7% and 74.6% CoNLL F1-Scores, respectively. However, each individual method generated improvements over the Base version. The overall results reported on B^3 and CEAF_e are considerably high, which means there is a good alignment between the predicted and gold entities and that a good percentage of mentions are being resolved to the right entity. MUC results are a little lower, which we assume is related to it disregarding singletons, which are annotated in Spanish data and represent a considerable amount of mentions, implying their correct prediction improves the other metrics. Our model has the best performance, in comparison with the previous works' results reported in Table 2.2 (+4.8% on CoNLL F1-Score).

For the monolingual training on Portuguese data, the model combining Base+DmUSE+DA had the best performance, with 64.1% CoNLL F1-Score. Individually, DA was the only method that did not improve the model; however, it improved the percentage of links and mentions predicted correctly from the gold data (higher Recall MUC and B^3 values). Results are lower compared to the ones obtained for Spanish data, as expected. That happens because Portuguese is a less resourced language. Despite that, we achieved promising results, reporting a better performance in comparison with the previous models' results presented in Table 2.2 (+14.4% on CoNLL F1-Score).

		MUC			B^3			$CEAF_e$			CoNLL
		P	R	F1	P	R	F1	P	R	F1	Avg F1
ES	Base	62.6	55.4	58.8	82.3	76.6	79.4	80.5	86.2	83.2	73.8
	+PCA	**65.8**	54.9	59.9	**83.5**	77.0	80.1	79.6	**87.6**	83.4	74.5
	+DmUSE	64.1	56.6	60.1	82.8	77.4	80.0	**81.1**	86.9	**83.9**	**74.7**
	+DA	64.1	56.4	60.0	81.5	77.4	79.4	80.6	86.5	83.5	74.3
	+PCA+DmUSE	62.7	56.4	59.4	82.2	77.1	79.6	81.0	86.0	83.4	74.1
	+PCA+DA	62.0	57.0	59.4	81.6	77.4	79.4	80.9	84.8	82.8	73.9
	+DmUSE+DA	63.0	56.8	59.7	82.3	77.4	79.8	**81.1**	86.0	83.4	74.3
	+PCA+DmUSE+DA	63.6	**57.4**	**60.3**	82.6	**77.9**	**80.2**	80.9	85.7	83.2	74.6
PT	Base	84.6	57.9	68.7	**87.8**	58.3	70.1	38.9	72.3	50.6	63.1
	+PCA	77.4	60.9	68.1	79.5	59.8	68.2	44.9	70.9	**55.0**	63.9
	+DmUSE	76.0	61.9	68.2	76.3	60.7	67.6	**45.7**	68.8	54.9	63.6
	+DA	77.5	**62.4**	69.1	74.8	**60.9**	67.2	43.9	67.2	53.1	63.1
	+PCA+DmUSE	79.9	60.3	68.7	81.9	60.0	69.3	42.4	70.8	53.0	63.7
	+PCA+DA	83.3	59.0	69.1	85.8	59.4	70.2	40.7	**72.9**	52.2	63.9
	+DmUSE+DA	**85.3**	59.3	70.0	87.0	59.4	**70.6**	39.8	**72.9**	51.5	**64.1**
	+PCA+DmUSE+DA	83.6	59.1	**69.2**	86.6	59.1	70.2	40.5	72.7	52.0	63.8

Table 4.2: Evaluation results for monolingual co-reference resolution on AnCora-CO-ES and Corref-PT datasets using gold mention boundaries.

In particular, we think that DmUSE improves the results on both languages not only due to the additional contextual information, but also due to providing a representation for out-of-vocabulary words existent in the test but not in the training data. Those words had no MUSE embeddings so previously they were represented with arrays of zeros.

Table 4.3 reports the results of the proposed cross-lingual model trained simultaneously with both Spanish and Portuguese data, using gold mention boundaries, and tested on each individual test set. The cross-lingual model succeeded in generalizing and performing co-reference resolution for both languages, offering competitive results for both Portuguese and Spanish. However, when compared to the monolingual approach, we cannot say that this generalization achieves better performance for either language. The only previous work exploring cross-lingual settings was the one presented by Cruz et al. (2018). They only reported results for the Portuguese test set, as they focused of direct transfer learning from Spanish to Portuguese. Our approach achieved better results in comparison with theirs (+18.9% on CoNLL F1-Score), which may be partially related to Portuguese data also being used for training.

On this cross-lingual training scenario, the model combining Base+DmUSE+DA had the best performance in the Spanish test data, with 74.9% CoNLL F1-Score. For the Portuguese test data, the best performance was achieved by the model combining Base+PCA+DmUSE, with 63.7% CoNLL F1-Score.

Tables 4.4 and 4.5 report the results for the Base model trained in the monolingual and cross-lingual scenarios, respectively, but using our mention detection function instead of gold mention boundaries. Regarding the mentions detected, the Recall values are high for both languages, around 85%, meaning 85% of the gold mentions were successfully detected. However, the Precision values are lower, meaning a lot of incorrect mentions were detected. For Spanish 48.3% of the mentions detected were right, while for Portuguese it was only 16.7%.

		MUC			B^3			$CEAF_e$			CoNLL
		P	R	F1	P	R	F1	P	R	F1	Avg F1
ES	Base	62.9	56.6	59.6	81.3	76.8	79.0	81.1	86.0	83.5	74.0
	+PCA	62.6	54.8	58.5	82.1	76.4	79.1	80.3	86.4	83.2	73.6
	+DmUSE	63.3	**56.7**	59.8	80.6	77.1	78.8	**81.5**	86.7	84.0	74.2
	+DA	63.6	56.1	59.6	83.1	77.0	80.0	80.8	86.6	83.6	74.4
	+PCA+DmUSE	65.5	56.2	60.5	83.0	77.2	80.0	80.7	**87.7**	**84.1**	73.9
	+PCA+DA	62.4	55.2	58.6	81.7	76.8	79.1	80.5	86.1	83.2	73.6
	+DmUSE+DA	**66.7**	56.6	**61.2**	83.0	**77.4**	80.1	79.8	87.2	83.3	**74.9**
	+PCA+DmUSE+DA	65.6	55.1	59.9	**84.9**	76.4	**80.4**	79.4	87.1	83.1	74.5
PT	Base	73.9	**64.0**	**68.6**	68.8	**62.7**	65.6	46.6	63.6	53.8	62.7
	+PCA	73.2	62.7	67.5	71.8	61.8	66.5	47.6	66.3	55.4	63.1
	+DmUSE	75.1	62.1	68.0	73.3	60.9	66.5	45.6	67.0	54.3	62.9
	+DA	**82.5**	55.8	66.6	**87.0**	56.6	68.5	38.5	**72.4**	50.3	61.8
	+PCA+DmUSE	74.4	63.3	**68.4**	72.7	62.2	67.0	**47.7**	67.1	**55.8**	**63.7**
	+PCA+DA	77.0	59.1	66.9	78.6	58.8	67.2	42.1	68.8	52.3	62.1
	+DmUSE+DA	80.8	58.5	67.9	81.8	58.5	68.2	39.8	69.8	50.7	62.3
	+PCA+DmUSE+DA	79.9	59.7	68.3	81.2	59.7	**68.8**	41.9	70.7	52.6	63.3

Table 4.3: Evaluation results for cross-lingual co-reference resolution on AnCora-CO-ES and Corref-PT datasets using gold mention boundaries.

		Mentions			MUC			B^3			$CEAF_e$			CoNLL
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg F1
ES	Base	48.3	85.3	61.7	64.8	43.3	51.9	42.9	63.3	51.2	31.7	77.2	45.0	49.4
PT	Base	16.7	84.8	28.8	60.7	31.0	41.0	15.5	37.7	22.0	3.6	62.4	6.8	23.3

Table 4.4: Evaluation results for monolingual co-reference resolution on AnCora-CO-ES and Corref-PT datasets using mention detection.

		Mentions			MUC			B^3			$CEAF_e$			CoNLL
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg F1
ES	Base	48.3	85.3	61.7	65.7	42.2	51.4	43.3	62.9	51.3	31.7	77.5	45.0	49.2
PT	Base	16.7	84.8	28.8	60.9	28.7	39.0	15.6	36.5	21.9	3.5	61.7	6.6	22.5

Table 4.5: Evaluation results for cross-lingual co-reference resolution on AnCora-CO-ES and Corref-PT datasets using mention detection.

As expected, the results for both scenarios are lower when compared to those obtained using gold mentions. Flaws in mention detection provoke error propagation to the co-reference resolution task, affecting its metrics. Similarly to the results obtained with the gold mentions, the cross-lingual model is capable of generalizing but reports slightly lower results in comparison with the corresponding monolingual models.

5

Conclusions and Future Work

Contents

This chapter summarizes the main conclusions from this dissertation, and suggests possible directions for future work on the co-reference resolution task.

5.1 Conclusion

In this work we presented a state-of-the-art neural co-reference resolution model for Portuguese and Spanish data, built on a previous one developed for English data - NeuralCoref. We tested additional methods to improve the model, namely post-processing for cross-lingual word embeddings and adding a mention feature based on cross-lingual contextual embeddings. Exploring another solution for the smaller amount of Portuguese resources for this task, we translated each training set to the other language, generating more co-reference annotated data. The feature with contextual embeddings provided some contextual information and representations for words outside the training data, showing slight improvements in the results. The data augmentation process by itself did not show improvements for Portuguese, despite achieving slight better results when combined with the other methods.

Our model was trained on a monolingual scenario for each language, using gold mention boundaries, and achieved a better performance in comparison with the existing ones, trained and tested on the AnCora-CO-ES and Corref-PT corpora. We also presented a cross-lingual variant of the co-reference resolution model, trained simultaneously on data from both languages. To the best of our knowledge, it is one of the first systems exploring cross-lingual learning with Spanish and Portuguese. The model succeeds in generalizing co-reference resolution for both languages and reports competitive results, in comparison with respective monolingual models. Additionally, we developed a mention detection function, capable of identifying candidate mentions on the AnCora-CO-ES and Corref-PT corpora, running experiments with it. The results obtained were considerably lower in comparison with the models using gold mentions, confirming that errors in mention detection affect the co-reference resolution task.

5.2 Future Work

Despite the interesting results, there is room for improvement in future work. As the model hyperparameters were not tuned, a simple future improvement would be to fine tune them. In the training process, the number of positive and negative (i.e. co-referent and non co-referent) instances is highly unbalanced towards the positive side, for both languages. Hence, a possible route is to explore undersampling, as proposed by Cruz et al. (2018).

Regarding the word embeddings, one possible option would be to replace the FastText embeddings by cross-lingual contextual embeddings such as ELMo or BERT, using only contextual representations. There are some methods available for their alignment, like the one presented by Schuster et al. (2019).

Alternatively there are also multilingual models already available, similar to the distilled mUSE. Focusing on the model, a new feature could be added. For a mention-pair, the contextual representations of its mentions (obtained from distilled mUSE) can be compared using cosine similarity, and that value can be added as a new mention-pair feature.

Another promising line of work is to improve the mention detection. Following recent research, the model can be adapted to jointly tackle mention detection and co-reference resolution, so those tasks share the same optimization goal, as in the work developed by Lee et al. (2017). We believe that an improvement to mention detection subtask would be reflected as an improvement on the co-reference resolution performance.

Bibliography

Artetxe, M., Labaka, G., and Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Attardi, G., Simi, M., and Dei Rossi, S. (2010). TANL-1: Coreference Resolution by Parse Analysis and Similarity Clustering. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 108–111. Association for Computational Linguistics.

Bagga, A. and Baldwin, B. (1998). Algorithms for Scoring Coreference Chains. In *Proceedings of the International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566. Granada.

Cai, J. and Strube, M. (2010). Evaluation Metrics for End-to-End Coreference Resolution Systems. In *Proceedings of the SIGDIAL 2010 Conference*, pages 28–36. Association for Computational Linguistics.

Clark, K. and Manning, C. D. (2016). Improving Coreference Resolution by Learning Entity-Level Distributed Representations. *arXiv:1606.01323*.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word Translation Without Parallel Data. *arXiv:1710.04087*.

Cruz, A. F., Rocha, G., and Cardoso, H. L. (2018). Exploring Spanish Corpora for Portuguese Coreference Resolution. In *Proceedings of the International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 290–295. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*.

Fonseca, E., Sesti, V., Antonitsch, A., Vanin, A., and Vieira, R. (2017a). CORP: Uma Abordagem baseada em Regras e Conhecimento Semântico para a Resoluçao de Correferências. *Linguamática*, 9(1):3–18.

Fonseca, E., Sesti, V., Collovini, S., Vieira, R., Leal, A., and Quaresma, P. (2017b). Collective Elaboration of a Coreference Annotated Corpus for Portuguese Texts. In *Proceedings of Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval)*, volume 1881, pages 68–83. CEUR-WS.

Fonseca, E., Vanin, A., and Vieira, R. (2018). Mention Clustering to Improve Portuguese Semantic Coreference Resolution. In *Proceedings of the International Conference on Applications of Natural Language to Information Systems*, pages 256–263. Springer.

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning Word Vectors for 157 Languages. *arXiv:1802.06893*.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*.

Kobdani, H. and Schütze, H. (2010). SUCRE: A Modular System for Coreference Resolution. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 92–95. Association for Computational Linguistics.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end Neural Coreference Resolution. *arXiv:1707.07045*.

Luo, X. (2005). On Coreference Resolution Performance Metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32. Association for Computational Linguistics.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013). Exploiting Similarities among Languages for Machine Translation. *arXiv:1309.4168*.

Mu, J., Bhat, S., and Viswanath, P. (2017). All-but-the-Top: Simple and Effective Postprocessing for Word Representations. *arXiv:1702.01417*.

Oliveira, H. G. (2012). *Onto.PT: Towards the Automatic Construction of a Lexical Ontology for Portuguese*. PhD thesis, University of Coimbra/Faculty of Science and Technology.

Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W.-t. (2018). Dissecting Contextual Word Embeddings: Architecture and Representation. *arXiv:1808.08949*.

Recasens, M., Màrquez, L., Sapena, E., Martí, M. A., Taulé, M., Hoste, V., Poesio, M., and Versley, Y. (2010). SemEval-2010 Task 1: Coreference Resolution in Multiple Languages. In *Proceedings*

of the International Workshop on Semantic Evaluation, pages 1–8. Association for Computational Linguistics.

Recasens, M. and Martí, M. A. (2010). AnCora-CO: Coreferentially annotated corpora for Spanish and Catalan. *Language resources and evaluation*, 44(4):315–345.

Reimers, N. and Gurevych, I. (2020). Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. *arXiv:2004.09813*.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747*.

Ruder, S., Vulić, I., and Søgaard, A. (2017). A Survey of Cross-lingual Word Embedding Models. *arXiv:1706.04902*.

Sapena, E., Padró, L., and Turmo, J. (2010). RelaxCor: A Global Relaxation Labeling Approach to Coreference Resolution. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 88–91. Association for Computational Linguistics.

Schuster, T., Ram, O., Barzilay, R., and Globerson, A. (2019). Cross-lingual Alignment of Contextual Word Embeddings, with Applications to Zero-shot Dependency Parsing. *arXiv:1902.09492*.

Silva, W. D. C. (2013). *Aprimorando o Corretor Gramatical CoGrOO*. PhD thesis, University of São Paulo.

Smith, N. A. (2019). Contextual Word Representations: A Contextual Introduction. *arXiv:1902.06006*.

Stylianou, N. and Vlahavas, I. (2019). A Neural Entity Coreference Resolution Review. *arXiv:1910.09329*.

Sukthanker, R., Poria, S., Cambria, E., and Thirunavukarasu, R. (2018). Anaphora and Coreference Resolution: A Review. *arXiv:1805.11824*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762*.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., and Hirschman, L. (1995). A Model-Theoretic Coreference Scoring Scheme. In *Proceedings of the Conference on Message Understanding*, pages 45–52. Association for Computational Linguistics.

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G. H., Yuan, S., Tar, C., Sung, Y.-H., et al. (2019). Multilingual Universal Sentence Encoder for Semantic Retrieval. *arXiv:1907.04307*.