TÉCNICO LISBOA

Neural Methods for Cross-lingual Sentence Compression

Frederico Infante Rodrigues

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Ricardo Daniel Santos Faro Marques Ribeiro

Examination Committee

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Bruno Emanuel da Graça Martins
Members of the Committee: Prof. Fernando Manuel Marques Batista

May 2018

Acknowledgements

I would like to thank Professor Bruno Martins and Professor Ricardo Ribeiro for all the patience, guidance and support throughout the development of this thesis. It has been a long journey with ups and downs but in the end due to the words of encouragement and motivation of my supervisors worth all the effort.

Secondly, I also would like to thank Professor David Matos and all the people from the INESC-ID who contributed with great advices, support and laughs.

Last, but not least I would like to thank my girlfriend and my family for all the support and for always being positive.

Frederico Infante Rodrigues

Resumo

A compressão de frases permite produzir uma frase mais pequena ao retirar a informação redundante, mantendo a gramaticalidade. Os sistemas actuais são baseados em redes neuronais que geram uma sequência binária de etiquetas para cada frase: se uma palavra se mantiver da frase original para a compressão é atribuída uma etiqueta com o número um, caso contrário é atribuída a etiqueta zero. Nesta tese, são propostas arquitecturas neuronais que tentam melhorar os sistemas actuais baseados em redes neuronais, especificamente é usado um método que permite gerar globalmente a melhor sequência de etiquetas para uma sequência de palavras, em vez de gerar independentemente como fazem os métodos actuais. Além de estratégias adicionais durante o treino do modelo é também considerado o uso de características sintácticas que podem ajudar a generalizar. Neste trabalho, a tarefa de comprimir frases é também extendida para uma configuração multilíngua que permite gerar compressões em Inglês e Português. A arquitectura proposta conseguiu resultados melhor ou iguais ao avaliar os modelos no mesmo conjunto de dados de teste que os sistemas actuais. Adicionalmente, ao avaliar os modelos nos dados em Português, a arquitectura com melhores resultados apenas usou as palavras de uma sequência, visto que o modelo que continha características sintácticas obteve resultados inferiores.

Abstract

Sentence compression produces a shorter sentence by removing redundant information, preserving the grammatically and the important content of the original sentence. This thesis proposes an improvement to the current neural deletion systems. These systems output a binary sequence of labels for an input sentence, the label one indicates that the token from the source sentence remains in the compression, whereas zero indicates that the token had been removed. Our improvement is the use of a method on the output layer which benefits the decoding of the best global sequence of labels for a given input. An auxiliary loss function is also considered as well as the incorporation of syntactic features which helps to capture grammatical relations. In addition, the sentence compression task is extended into a cross-lingual setting where the models are evaluated on English and Portuguese. The proposed architecture has achieved better or equal results than the current systems, validating that the model benefited from the modification in both languages.

Palavras Chave
Keywords

Palavras Chave

Compressão de Frases

Sumarização

Redes Neuronais profundas

Multilingue

Vetores de Palavras

Keywords

Sentence Compression

Summarization

Deep Neural Networks

Cross-lingual

Word Embeddings

Contents

List of Figures

List of Tables

1 Introduction

Sentence compression is a Natural Language Processing (NLP) task that returns a shorter version of a sentence, maintaining its readability and the most important information.

This task can be applied in news digests, subtitle generation and automatic summarization systems, which produce summaries from a document at a sentence level benefiting from the use of these compression systems.

A compression system is often formulated as deletion-based, which indicates that the output is a sequence of labels, namely zeros and ones, representing if a token was deleted or not from the source sentence.

Tree-based methods are often used to sentence compression, a sentence is parsed by a dependency tree and the process of compression is done by removing dependency edges from the tree. Although, this approach often work, if a sentence is ambiguous and have multiple possible parses, the compression could lead to grammatical errors.

Thus, instead of using the parsed tree as output the former systems prefer to use it as a feature. Another popular approach is to formulate sentence compression as an Integer Linear Programming (ILP) problem using dependency tree features as constraints as well as another linguistic features to ensure the grammaticality of the output compression.

Recent work, focus on deep neural networks which even without any linguist features generate compressions grammatically correct. However, these system also incorporate syntactic features to generate better compressions.

These tasks are mostly used in a monolingual setting because each language has its grammatical rules and it may be difficult to generate readable sentences across languages. Therefore is necessary to learn good cross-lingual representations that project different languages into a shared space.

In order to evaluate this method on a cross-lingual setting, in this work a set of experiments is performed on two languages, namely, English and Portuguese. The datasets used in the experiments are both publicly available and the metrics used were the same as in previous

neural approaches.

1.1 Thesis Proposal

Current compression systems are based mostly in recurrent neural networks which benefit sequences. It is important to improve the current systems by producing compressions which are better grammatically and understandable.

The proposed architecture is based on the current LSTM deletion-based architecture systems (Filippova et al., 2015; Wang et al., 2017). These systems output a sequence of binary labels for an input sentence, meaning if a token remains or not in the compression version.

The main modification refers to the final layer where these systems often use a method that decodes the best label independently. Instead, this work proposes a layer that benefits the whole sentence and decodes the best global sequence of labels for an input sequence taking into account possible correlations between neighbours making the output globally consistent.

Inspired in the previous work (Filippova et al., 2015; Wang et al., 2017), syntactic features are incorporated into the model which should help the model to generalize better. The following features are considered: POS embeddings of each word, the embedding of the parent word in the dependency tree as well as the respective POS, and the dependency relation embedding relative to the head of each word.

Additionally, an auxiliary loss function considering the log frequency of each word is also used. This approach helps the system to account for rare words predicting the label and the log frequency for each word jointly.

In order to prevent overfitting, data augmentation is also used during the training on the English corpus where new sentences are created from the ground-truth compression. This method is possible because there are spans of words in the sentence that are not taking into account in the compression, making it possible to construct a new sentence-compression pair.

Furthermore, this work also contributes with a new corpus with 799 sentence pairs for sentence compression in Portuguese.

Finally, the sentence compression task is extended into a cross-lingual setting where the models proposed are evaluated on two sources of data.

1.2 Structure of the Document

The rest of this document is organized as follows. Chapter 2 presents fundamental concepts and related work in the sentence compression area. Then, Chapter 3 describes the different approaches to compress sentences. Chapter 4 presents the experimental setup: the datasets used, evaluation metrics, experimental settings and the results. Finally, Chapter 5 concludes this document by summarizing the main findings of this work, and highlighting possible directions for future research.

2 Concepts and Related Work

This chapter presents in the Section 2.1 the fundamental concepts and in Section 2.2 is described the related work on sentence compression discussing the different methods used

2.1 Fundamental Concepts

In this section, an overview of the Vector Space Model, term frequencyâinverse document frequency (TF-IDF), and Cosine Similarity are provided on Section 2.1.1. In Section 2.1.2 it is explained what is a word embedding and the current pre-trained embeddings used. In Section 2.1.3 it is described what recurrent neural networks are and why are they important in NLP. Finally, in section 2.1.4 is described the method used to decode globally the best sequence of labels for a given sequence. [FIX]

2.1.1 Vector Space Model, TF-IDF, and Cosine Similarity

The Vector Space Model (VSM) is an algebric model for representing textual documents as vectors in a high dimensional space where each dimension corresponds to a term ocurring over the document collection (e.g., a word or an n-gram). Each component of the vector is denominated a term weight and different approaches can be considered for computing the importance of terms in the context of representing a document's content.

The most simple way of calculating term weights is by a binary value, with one indicating that the term occurred in the document and zero otherwise. A second alternative is term frequency, which just places in each component of the vector the number of times the corresponding term occurred in a specific document.

The most popular term weighting scheme is Term Frequency $\times$ Inverse Document Frequency (TF-IDF), telling how a word is important to a document given a set of documents. Each word is assigned a weight that is obtained by multiplying two components: TF and inverse term IDF. TF is a function of the number of times that the word appears in a document and

the IDF is based on the number of documents in the collection containing the word, reducing the weight of terms that appear often and raising the weight of terms that rarely appear.

In this task, the vectors are usually built from sentences to represent only one document, so the meaning of the weighting schemes is slightly different. In this case, TF-IDF shows how important is a word in a sentence, given the other sentences that form a document. Given a document $D = \{S_1, S_2, \ldots, S_N\}$ with N sentences, the vector representation of a sentence is given by $S_i = \{w_{i1}, w_{i2}, \ldots, w_{im}\}$, where w_{ik} is the weight of the term k in the sentence S_i, and is calculated according to:

$$\mathrm{w_{ik}} = \mathrm{f_{ik}} \times \log\left(\frac{n}{1+n_k}\right) \tag{2.1}$$

In Equation 2.1, f_{ik} represents the frequency of the term t_k in the sentence S_i.

A traditional similarity measure is the cosine of the angle between the vectors representing two sentences, S_i and S_j, and is calculated according to Equation 2.2.

$$\mathrm{sim_{cos}}(S_i, S_j) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} w_{jk}^2}\sqrt{\sum_{k=1}^{m} w_{ik}^2}} \tag{2.2}$$

One of the problems of using the VSM occurs for example when two sentences (i.e., two vectors) mean the same but they do not have any words in common. Similarity would be zero because the vectors are orthogonal, even though the concepts are related. Vector embeddings approaches emerged to overcome this problem, capturing syntactic and semantic information about each word.

2.1.2 Word Embeddings

Word2vec is a popular predictive model for learning word embeddings from raw text (Mikolov et al., 2013). Word2vec introduced two model architectures for learning these representations: the Skip-Gram, and the Continuous Bag-of-Words (CBOW). Both models are shallow, being composed by only a neural network of two-layers. A neural network is usually composed by a set of neurons which are weighted interconnected by multiple layers. There are three types of layers: input, hidden and output layer. In order to produce an expected output is necessary to train the network by updating the weights: this is done through the well know backpropagation algorithm (Rumelhart et al., 1986).

While CBOW predicts a word given its context, the Skip-Gram model predicts the context of a given word, as illustrated in Figure 2.1.

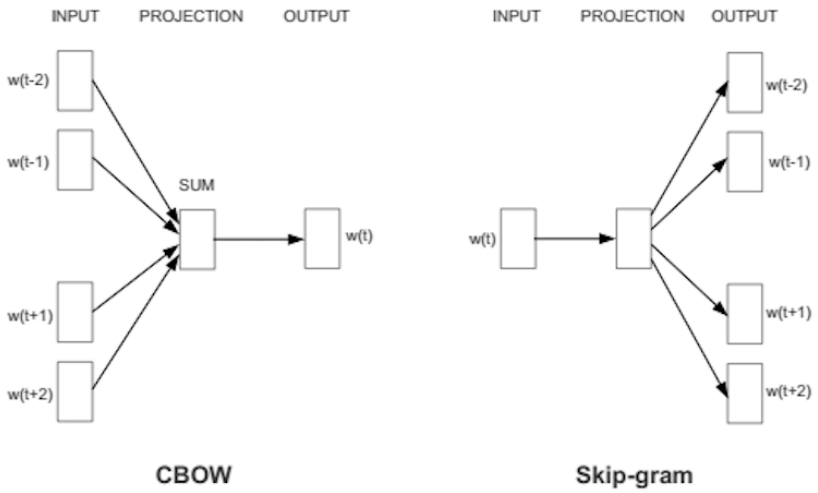

Figure 2.1: The CBOW and Skip-Gram architectures, adapted from Mikolov et al. (2013).

In the first architecture, the idea is to predict a word w_t based on n words before and after it. It is called a bag-of-words model because the order of the words it is not important at the projection layer. The following objective function is maximized, feeding the model with n words around the target word, w_t.

$$\frac{1}{T}\sum_{t=1}^{T}\log \quad p(w_t|w_{t-n},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+n}) \tag{2.3}$$

The Skip-Gram model is trained to predict surrounding words given the current word. Each word is trained to maximize the log probability of neighboring words in a corpus, according to the following objective:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log \quad p(w_{t+j}|w_t) \tag{2.4}$$

In Equations 2.3 and 2.4 , T is the number of words in the training set, and c is the size of the training context.

This pre-training embeddings represent individual words as vectors that encode semantic meaning (i.e., the output layer illustrated in Figure 2.1). These word representations can be used to compute similarity operations between words, or they can be combined in order to build document representations (e.g., by averaging the word embeddings, or through more complex procedures).

Bojanowski et al. (2016) proposed fastText embeddings which learn word representations from character n-grams and represent a word as a sum of n-gram vectors. Their word embeddings are learned by using the Skip-Gram model, which was trained on Wikipedia data. This type of embeddings benefit a internal structure of a word, representing better words that are misspelled or out of vocabulary words.

Recently, Mikolov et al. (2018) used a combination of three strategies to improve the learning of the word representation: position-dependent weighting, phrase representations and subword information. This model rely on the CBOW model, which outputs a word representation according to its context. A context vector is represented by the average of word vectors contained in it, but the latter models did not take into account the position of the words. Therefore, they propose to use an additional position vector that encodes the position of each word present in the context vector. Thus, the context vector is the average of context words reweighted by their position vectors.

Another strategy used to improve the word embeddings is phrase representations. For example *New York City* or *New York University* are replaced by unique tokens. These phrases are formed when in the training corpus words appear frequently together, and infrequently in other contexts. However, to prevent the formation of too many phrases they are created using the following score function:

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i, w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)} \tag{2.5}$$

where δ is used as discount coefficient preventing phrases with infrequent words to be formed below a certain threshold. Finally, the use of subword information follows the idea of Bojanowski et al. (2016) where a word is represented as a sum of the character n-grams as described above.

Grave et al. (2018) extend the work of Bojanowski et al. (2016) by using different parameters and two sources of data, learning better word representations for 157 languages other than English. They performed a variation of some of the parameters of the default model of fastText. Instead of using Skip-Gram they used CBOW while training the model on Wikipedia and on the crawl data, using 10 epochs. Additionally, 10 negative examples were used instead of the 5 default and set the n-gram size to 5.

2.1.3 Recurrent Neural Networks

Another important concept to understand is how deep neural networks can be used in tasks related to representing and classifying textual contents, leveraging word embeddings.

Recurrent Neural Networks (RNNs) are one of the most commonly used types of neural networks when dealing with language data because their structure supports processing sequential information, e.g. sequences of words. They are recurrent because each cell can be unrolled through time and the same operation is applied to each element of the sequence.

Figure 2.2 represents an unrolled RNN where at time step t, the variable x_t represents the input, A the hidden state, and h_t represents the output. Note that Figure 2.2 shows outputs at each time step, but this may not be necessary depending on the type of task.

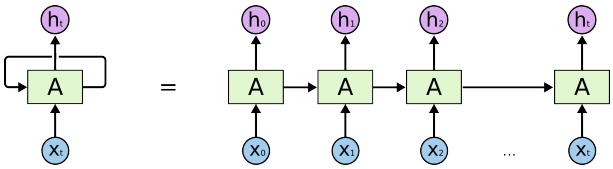

Figure 2.2: An unrolled Recurrent Neural Network, adapted from Olah (2015)

The hidden state is an important feature because it captures the information along the sequence. It is calculated based on the previous hidden state and the input at each time step. This way, the final hidden state contains the information about all the sequence that was processed.

The training of this network is similar to training a simple neural network, leveraging an adapted version of backpropagation, that it is called backpropagation through time (BPTT).

One of the problems of conventional RNNs is the vanishing gradient problem, because if we are modelling a long sequence of words, it is necessary to update weights by propagating the cost until the initial state. This leads to multiplication of small gradients over the sequence. To overcome this effect, emerged LSTMs and GRUs.

Long Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are a type of RNN which can learn long-term dependencies and get around the problem of vanishing gradients. One special feature of LSTMs is the memory C_t, which allows the model to manipulate the information that goes in and out of each cell through a mechanism of gates. The formula is defined in Equation 2.6:

There are essentially three gates involved in LSTMs: input i_t, forget f_t and output gate o_t. These control the cell state, i.e. the memory. The three gates have different roles, for instance, an input gate decides which values are going to be updated (Equation 2.7):

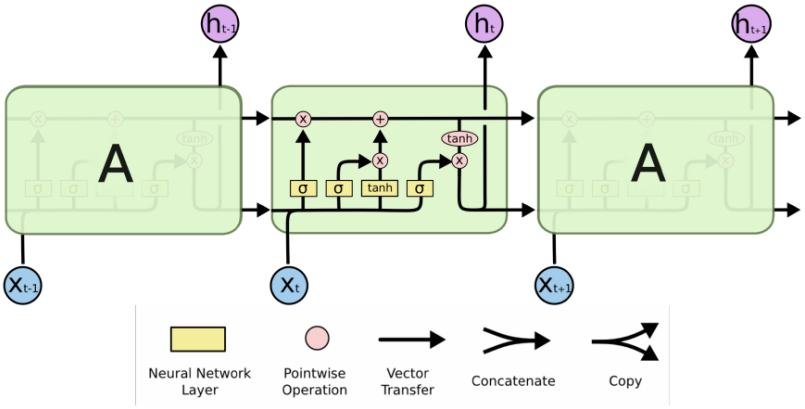

Figure 2.3: Long Short Term Memory, adapted from Olah (2015)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{2.7}$$

A forget gate is responsible for deciding how much information is going to be retained considering the current input and the output of the previous layer (Equation 2.8):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

Finally, the output layer decides what to output based on the cell state(Equation 2.9):

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{2.9}$$

In order to increase the information captured, BiLSTMs emerged so that each element has a state that is composed by a backward state and a forward state, which are computed with two stacked LSTMs, one is fed with an input sequence and the other receives the input sequence in reverse. Thus, the output at each time, depends from previous and future elements. Also, to increase high order representation, BiLSTMs can be stacked in layers.

Another type of RNN, named Gated Recurrent Unit (GRU) was introduced by Cho et al.

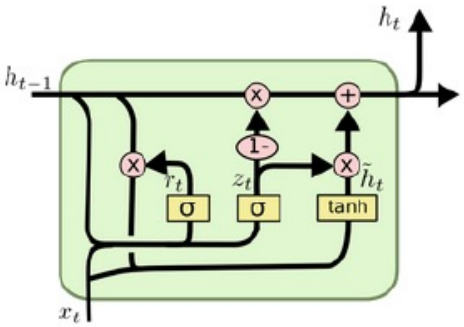

Figure 2.4: Gated Recurrent Unit, adapted from Olah (2015)

(2014). This model is less complex and also computationally less expensive than LSTMs, due to the fact that the forget and input layer are a single *update gate*. Moreover, the cell state and hidden state are merged. The previous simplifications also result in the simpler model illustrated on Figure 2.4.

Mathematically, the Equations 2.10 represent the update gate z_t, reset gate r_t, and the ouput hidden state h_t.

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{2.10}$$

2.1.4 Conditional Random Fields

A Conditional Random Field (CRF) is a method often applied in NLP for sequence labeling tasks such as named entity recognition (NER) and part-of-speech (POS). It is important to find globally the best sequence for a given input than decode the best label independently, since it helps to consider correlations between labels in neighbourhoods.

Consider an input sequence $x_1, \ldots, x_n$ and an output sequence of labels $y_1, \ldots, y_m$, CRF models the conditional probability of $P(y|x)$ using a feature map $\phi(x_1, \ldots, x_n; y_1, \ldots, y_m)$. Each feature function has a weight w associated, allowing in this way to score each function according to how well a sequence of labels fits a given sequence input. The score function is defined as :

$$\text{score}(y, x) = w \cdot \phi(x_1, \ldots, x_n; y_1, \ldots, y_m) \tag{2.11}$$

The score function is transformed in a probability by normalizing and exponentiating:

$$p(y|x; w) = \frac{e^{w \cdot \phi(x,y)}}{\sum_{y'} e^{w \cdot \phi(x,y')}} \tag{2.12}$$

where y' ranges over all possible output sequences. Learning the parameter w, is usually done by maximum likelihood learning

$$L(w) = \sum_i logP(y|x; w) \tag{2.13}$$

Finally, when the system receives an input sequence, the naive way to calculate the best label sequence is to go through every possible labeling and choose the one that maximizes the probability $p(y|x)$. However, this approach can lead to an exponential complexity, so it is used the Viterbi algorithm (Forney, 1973).

2.2 Related Work

In this section, some of the methods to approach sentence compression task are presented. Section 2.2.1 describes the methods based on syntactic parsing. Section 2.2.2 focus on the methods based on sequence classification and Section 2.2.3 the methods based on deep neural networks.

2.2.1 Methods Based on Syntactic Parsing

McDonald (2006) presented a deletion-based discriminative online learning model for sentence compression that has as input a set of features: POS tags and syntactic features provided by a dependency and a constituency parser. This is a supervised learning approach which is trained on a corpus of sentence/compression pairs. The features are calculated over adjacent words in the compression sentence. Regarding POS features some examples are: POS bigrams for adjacent words, POS of dropped tokens, a feature indicating if the two adjacent words in the compressed sentence were in the original sentence, and finally, a feature to represent brackets in a text because they often delimit redundant information.

The other set of features consist in deep syntactic inputs. Every sentence is parsed by a dependency and a constituency parser tree. First, for every dropped word in the compression sentence a feature is added indicating the POS of the parent word. The constituency parser is useful to add another feature indicating the context from where a node was dropped. Therefore, this set of features allows to encode the properties of their syntactic relation concerning the full sentence.

The learning process to learn the feature weights w was done through Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003). Each iteration of this algorithm tries to maximize the score $s(x, y)$. The loss was defined using the number of words that were dropped or retained incorrectly in the compressed sentence relative to correct one.

The score function is defined in Equation 2.14.

$$[h!]s(x, y) = \sum_{j=2}^{|y|} w \cdot f(x, I(y_{j-1}), I(y_j) \tag{2.14}$$

The results of this technique show benefits from the use of a rich dependency feature set to help optimize a function related to compression.

Niklaus et al. (2017) described a syntax-based sentence compression framework. This approach does not delete constituents because this could lead to a loss of information. By analyzing hundreds of sentences, the authors determined a set of constituents that only provide background information. The framework has as an input sentence that is parsed through a constituency parser tree, identifying possible constituents that only provide supplementary information, such as adjective and adverb phrases delimited by punctuation or phrases offset by commas.

The system follows a three-stage approach. First, those constituents that only provide auxiliary information are separated from the main sentence. Then is constructed a context sentence around the auxiliary information in order to be grammatically correct.

The authors do not report any results but stated that their framework simplifies an input sentence based linguist features, instead of deleting constituents that only provide non-essential information, these are embedded into new context sentences. In other words, the system breaks a complex syntax sentence into shorter sentences.

2.2.2 Methods Based on Sequence Classification

Clarke and Lapata (2008) formulate the sentence compression task as a binary decision for each word in the source sentence considering if a word should remain or not in the compressed version. They consider this task as an optimization problem and solve it using ILP.

The objective function is based on a trigram model and a significance score. The former takes into account the grammaticality of the compressed sentence and the latter verifies if the words that remained in the compression are considered important.

The significance score highlight important content word and the trigram model is represented by the sum of all possible trigrams in a compression and is given in Equation 2.15

$$\sum_{i=1}^{n} p_i \cdot \log p(w_i|start) + \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} x_{ijk} \cdot \log p(w_k|w_i, w_j) + \sum_{i=1}^{n} q_{ij} \cdot \log p(end|w_i, w_j) \quad (2.15)$$

where, $p_i = 1$ if w_i starts the compression , $q_{ij} = 1$ if the sequence w_i, w_j ends the compression, $x_{ijk} = 1$ if the sequence w_i, w_j, w_k is in the compression , $y_i = 1$ if w_i is in the compression.

In order to ensure the grammaticality and content of a compression, a set of constraints are defined to force some words to be kept in the output: if a modifier is included in the compression, the word associated with it also needs to be included. In addition, when a verb is present in the compression sentence, the subject and object must be in the included in the output.

Moreover, a negation constraint is also used when the word *not* is connected with another word: both need to be in the compression. Finally, another constraint is when personal pronouns are in the source sentence they must be kept in the compression.

The authors evaluated their system on a new corpus. They use 82 news articles from the Washington Post, the LA Times, and the British National Corpus (BNC) and instructed annotators to delete individual tokens from each sentence keeping the compression grammatical and preserving the most important information.

They evaluated their system automatically using a F-score measure based on the grammatical relations between the the ground truth compressions and the ones predicted by the system.

Filippova and Strube (2008) presented a unsupervised approach that relies on a dependency tree. The compression task is formulated as an ILP problem and the best sub-tree is the one

with highest score from the objective function. Finally, the words of the compressed sub-tree are presented in the same order as the source sentence.

The algorithm to compress a sentence is divided in 3 stages: tree transformation, tree compression and tree linearization.

First, a sentence is parsed by a dependency tree which suffered some transformations. The tree is transformed in a dependency graph, in order to guarantee the grammatically when pruning. The transformations are the following: auxiliary, determiner, preposition, negation and possessive nodes are collapsed with their heads, prepositional nodes are removed and placed as labels on the edge from their head to the respective noun, and the root node is connected with every inflected verb.

After these transformations the compression task is formulated as an optimization problem which is solved by using ILP. The goal is to find the sub-tree which gets the highest score on the objective function.

$$f(X) = \sum_{x} x^{l}_{h,w} \cdot P(l|h) \cdot I(w) \tag{2.16}$$

This function is composed by the probability of dependencies $P(l|h)$, where l stands for the edge label, and h is the head word w. In addition, word importance $I(w)$ also contributes for the objective function. If the dependency is preserved $x^{l}_{h,w} = 1$, otherwise $x^{l}_{h,w} = 0$.

The word importance formula (Equation 2.17) is based on the work of Clarke and Lapata (2008)

$$I(w_i) = \frac{t}{N} \cdot f_i \log \frac{F_A}{F_i} \tag{2.17}$$

w_i is the topic word (either noun or verb), f_i is the frequency of w_i in the sentence, F_i is the frequency of w_i in the corpus, and F_A is the sum of frequencies of all topic words in the corpus. t is the number of clause nodes above w and N is the maximum level of embedding of the sentence w belongs to.

This ILP problem is subject to structural and syntactic constraints in order to produce sub-trees grammatically correct.

Martins and Smith (2009) performed summarization using a joint model for compression and extraction using ILP to address the problem, which has the downside of having a highly

computational cost. Thus instead of using the same approach by Clarke and Lapata (2008) which also modelled compression as an ILP, with the optimization requiring $O(N^2)$ variables and constraints.

In order to reduce that complexity, Martins and Smith (2009) provided a new formulation for sentence compression based on the output of a dependency parser tree leading to only $O(N)$ variables and constraints. Concerning the formulation of extractive component, the authors also reduce from a complexity of $O(M^2)$ to $O(M)$ using a maximal marginal relevance based framework.

The joint model combines the extraction and compression scores into a global optimization problem maximizing it through ILP, using MIRA Crammer and Singer (2003) to learn the model parameters.

2.2.3 Methods Based on Deep Neural Networks

The use of Recurrent Neural Networks (RNNs) is common in NLP problems: Filippova et al. (2015) used a Long Short Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997), designed to solve the problem of vanishing gradients through a gating mechanism, and to remember long-distance dependencies from an input sequence.

The general idea is to feed the network with a sentence and translate it into a sequence of zeros and ones, where zero represents a token that is deleted and one represents a token that remains in the compressed sentence.

The network architecture, which is shown in Figure 2.5, is formed by three stacked LSTM layers in order to learn better representations from input, interleaved with dropout to prevent overfitting. The output layer is a softmax classifier that predicts at each timestep if a word is retained, deleted, or end-of-sentence (EOS).

The authors presented three ways of representing an input with the follow architectures: LSTM, LSTM+PAR and LSTM+PAR+PRES.

The LSTM architecture has 259 dimensions, where the first 256 represent the word embedding from the Word2Vec (Mikolov et al., 2013) model of the current word. The last three dimensions represent an one-hot vector with the label of previous word (0, 1, or EOS). For the other two architectures, the authors first parsed an input sentence with a dependency parser, this way it is possible to know the parent of the current word.

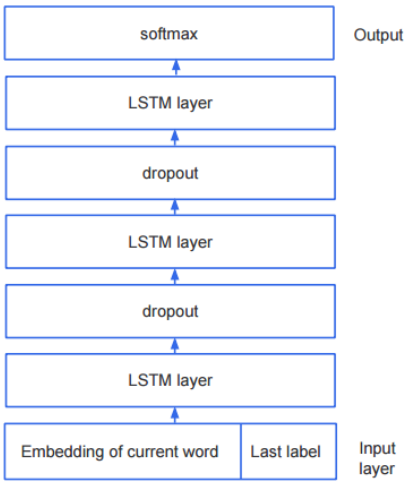

Figure 2.5: Archictecture of the network, adapted from Filippova et al. (2015).

Therefore, LSTM+PAR has 515 dimensions because it additionally represents a word embedding vector with 256 dimensions for parent word plus the previous mentioned 259 dimensions. The last one, LSTM+PAR+PRES, has an input vector with 518 dimensions. This architecture includes the previously mentioned embeddings, but with 3 more dimensions: a dimension indicating whether the parent word has already been seen and kept in the compression, another dimension if the parent word has already been seen but discarded, and a last dimension indicating if the parent word comes later in the input.

At decoding, enumerating and scoring all set of output sentences is exponential in the length of the input sequence, so it was used a beam-search algorithm for compressing.

This model was trained in a collected dataset of about two million parallel sentence-compression instances from news, following the method of Filippova and Altun (2013). The manual evaluation was done in the first 200-sentences of the test set and the first 1000 sentences were used for automatic evaluation.

Automatic evaluation used two metrics: per-sentence accuracy, which takes into account how many compressions could be fully reproduced, and word-based F1-score that computes recall and precision regarding to the words kept in golden collection references and the generated compression, whereas manual evaluations used readability and informativeness as metrics.

From the manual evaluation, it is showed that compressed sentences produced by the simplest LSTM architecture are more readable and informative than the others, showing that there is no benefit in introducing syntactic information at the input, achieving results better than the approach proposed by McDonald (2006).

Wang et al. (2017) proposed two major changes to the model of Filippova et al. (2015), introducing syntactic features into the model and proposing to use ILP to find an optimal combination of the labels.

Instead of using a single LSTM, Wang et al. (2017) use a BiLSTM to process a sequence of words in both directions with a softmax classifier in the end to predict each label independently. At the input layer word embeddings, POS and dependency relation embeddings are concatenated. They propose the addition of these syntactic features because they help to generalize better and because the model was tested in a cross-domain setting.

In addition, they propose to use ILP for finding an optimal combination of labels for an input sequence. The objective function is based on the probability estimated by the BiLSTM model and the depth of word in the dependency parse tree. They further introduce some constraints related to the length of a the compression and the syntactic structure.

2.3 Overview

In this section it was discussed some concepts that were needed to understand this work. Pre-trained embeddings are used to represent words as vectors containing semantic and syntactic information. These are used as input of the recurrent neural networks that are often used when dealing with NLP problems due to their recurrent structure that model a sequence of words. Since, this task is a sequence labeling problem, a CRF is used to decode globally the best sequence of labels. In addition, an overview of the methods to handle this task is also discussed.

3 Sentence Compression

This chapter details the different approaches that have been adopted to deal with sentence compression and also how this task is extended into a cross-lingual setting. In Section 3.1 it is described the base model which only has as input word embeddings. In Section 3.2 is described how syntactic features are introduced into the model and the use of an auxiliary loss function during training. Moreover, in Section 3.3 it is explained the extension of the sentence compression task into a cross-lingual setting. Finally, in Section 3.4 an overview of the methods applied is summed up.

3.1 Base Model

In this work, the sentence compression task is considered a deletion-based system. The output of the system is a binary sequence of labels which represent if a word is deleted or not from the source sentence. The neural network architecture proposed is inspired in previous work by (Filippova et al., 2015) and (Wang et al., 2017) which also consider a deletion-based system.

The main architecture is based on a combination of two stacked BiLSTMs followed by a CRF layer which outputs a binary sequence of labels given a source sentence.

A BiLSTM is used due to its recurrent structure which is good to process sequential information, and because it can process a sequence in two ways: from start to end, and in reverse. Capturing contextual information in both directions.

The CRF layer allows to decode globally the best sequence of labels considering correlations between neighbours instead of the use of a softmax which decodes the best label for each word independently.

An input sequence of words is denoted as $s = (w_1, w_2, \ldots, w_n)$. Each w_i belongs to a vocabulary, $w_i \in V$ which contains English and Portuguese words. There are words in s that may be deleted, thus the compressed sentence is represented by a sequence of binary labels $y = (y_1, y_2, \ldots, y_n)$, where $y_i \in 0, 1$. Here $y_i = 0$ represents a token that was deleted from the

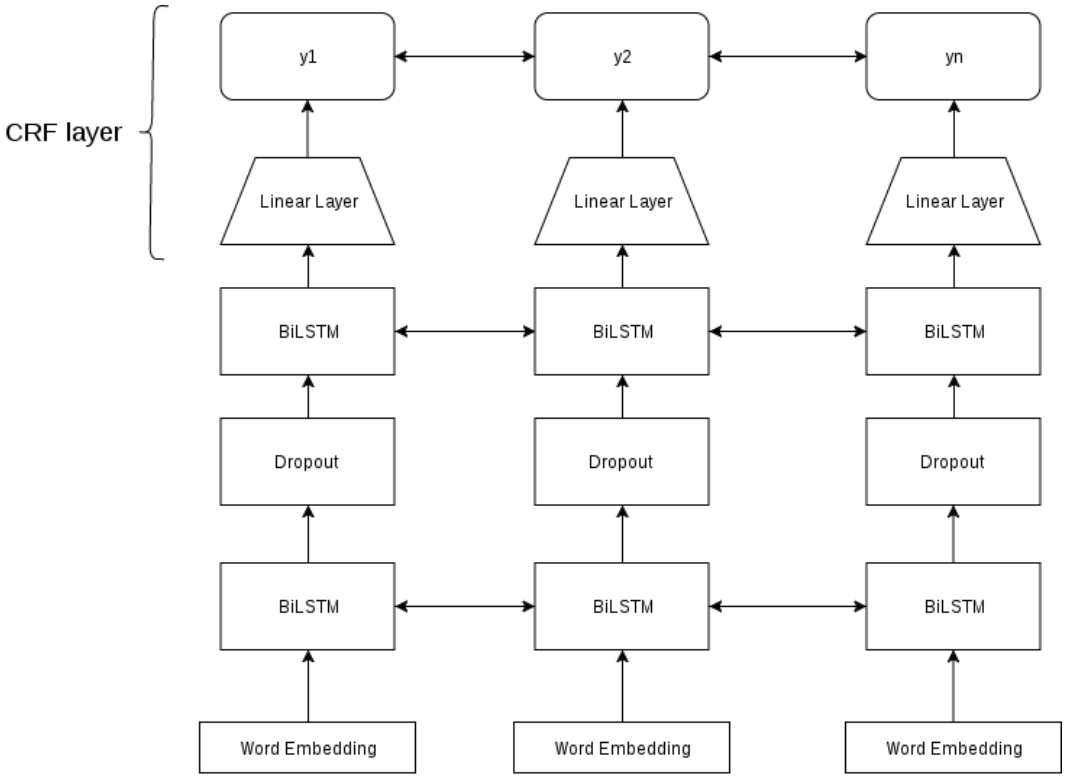

Figure 3.1: Architecture of the network.

original sequence and $y_i = 1$ indicates that the token remains. The input sequence maximum length is 80 tokens.

Each w_i is mapped to a 300-dimension pre-trained embedding, using fastText embeddings (Mikolov et al., 2018) in which a word is represented by a sum of representations of character n-grams. This approach allows a better a representation of words that were misspelled and rare word as well.

These embeddings are fed into a BiLSTM, one at a time, processing a sequence from left to right and in reverse, capturing contextual information from both directions. The hidden vectors go through a dropout layer to prevent overfitting (Srivastava et al., 2014), and are fed into another BiLSTM. The concatenated output of the last BiLSTM is mapped with a dense layer and then a linear-chain CRF maximizes the best sequence of labels for each input sequence. This architecture is represented in Figure 3.1.

On sequence labeling tasks such as POS and NER (Huang et al., 2015; Ma and Hovy, 2016) this architecture has achieved state-of-art results (Reimers and Gurevych, 2017).

3.2 Incorporation of Syntactic Features and an Auxiliary Loss

The incorporation of syntactic features into these models has shown improvements in the results of the neural network systems (Filippova et al., 2015; Wang et al., 2017). Considering a set of Universal POS tags, illustrated on Table 3.1. It was performed POS tagging on each input sentence which has an embedding vector associated that need to be learned during training.

Open class words	Closed class words	Other
	ADP	
ADJ	AUX	
ADV	CCONJ	PUNCT
INTJ	DET	SYM
NOUN	NUM	X
PROPN	PART	
VERB	PRON	
	SCONJ	

Table 3.1: Universal POS tags.

For each sequence, it was also performed dependency parsing. Each word is replaced by the dependency relation connecting to its head. During the training the weights of this embedding are learned.

In the input layer, there were considered different syntactic features to incorporate in the model, also it were tested different combinations between these features in the input of the model:

- Word + POS embeddings;
- Word + POS + Dependency relation embeddings;
- Word + POS + Parent word + POS parent embeddings;
- Word + POS + Parent word + POS parent + Dependency relation embeddings.

Furthermore, following the work of (Plank et al., 2016) an auxiliary loss was introduced in the model while training. The model jointly predicts the label and the log frequency of the word. The intuition behind this additional loss in the model it is because it benefits the handling of rare tokens.

3.3 Cross-lingual Setting

In this work, the proposed models are extended into a cross-lingual setting. Cross-lingual systems need to perform well across languages, thus learning cross-lingual embeddings is essential to represent different languages in a shared space. This way words from different languages but meaning the same are close to each other.

There are different approaches (Ruder et al., 2017) to learn these embeddings:

- Monolingual mapping: Initially monolingual word embeddings are trained on a large monolingual corpora individually and after that a linear mapping between the source and the target language allows to map unknown words into a shared space.
- Pseudo-cross-lingual: The embeddings are trained in a corpus with mixing contexts of different languages. These cross-lingual contexts allow the learned representations to capture cross-lingual relations.
- Cross-lingual training: These embeddings are trained in a parallel corpus, meaning that similar words in different languages are going to be closer in the semantic space.
- Joint optimization: The models are trained on parallel and jointly optimize a combination of monolingual and cross-lingual losses.

This work uses the approach of Conneau et al. (2017), which relies on monolingual mapping. Based on two distributions of monolingual data, the main objective is to align them in shared space.

Their method uses adversarial training to learn a linear mapping from a source to a target space. While a discriminator is used to distinguish which distributions the embeddings come from, the generator tries to fool it. An automatic synthetic parallel dictionary is built from the adversarial training. For this work it was used the publicly[1] released bilingual dictionary of Portuguese-English pairs.

Instead of using directly the 200-dimension embeddings provided by the authors, we used their tools to align in a supervised way the English and Portuguese embeddings into a shared space of 300 dimensions, which resulted in a better aligned space.

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

Finally, the models used in a cross-lingual setting were trained with English and Portuguese word embeddings that were aligned in the same subspace.

3.4 Overview

This chapter presented the different approaches to deal with sentence compression. From the base model which only used word embeddings as input to the incorporation of syntactic features and also an auxiliary loss. In this work, the application of different settings into this task is explored. In addition, this task was extended into a cross-lingual setting using cross-lingual embeddings.

4 Experimental Evaluation

This chapter presents the experimental setup in Section 4.1 and the results in Section 4.2.

4.1 Experimental Setup

In Section 4.1.1 both datasets are described, Section 4.1.2 presents the process of data augmentation, and in Section 4.1.3 the metrics to evaluate the system. Finally, in Section 4.1.4 it is explained the setup of the model.

4.1.1 Datasets

In order to evaluate the proposed models, two sources of data in different languages (English and Portuguese) were used.

The English dataset is publicly[1] released by Filippova and Altun (2013), it contains 200,000 sentence-compression pairs for training and 10,000 sentence pairs for testing. This corpus was built by collecting news articles in English: for every article the headline and the first sentence were extracted because they are known to be semantically similar.

Although not all pairs are suitable for compression, in order to filter the relevant pairs, it is important that the headline is a sub-sequence of words from the first sentence and these pairs match lemmas of content words (nouns, verbs, adjectives, and adverbs).

The first sentence is parsed by a dependency parser and then transformed in a dependency graph following a set of rules. For example, auxiliary, determiner, preposition, negation and possessive nodes are collapsed with their heads. Once all the nodes in the transformed graph match the content words from the headline, a compression for the sentence is generated when the minimum sub-tree covering these nodes is found (Filippova and Altun, 2013).

The Portuguese dataset is publicly[2] released by Almeida et al. (2014). This corpus contains

[1] `https://github.com/google-research-datasets/sentence-compression`
[2] Available at `http://labs.priberam.pt/Resources/PCSC.aspx`

801 documents split in 80 topics. Each topic has two human-made summaries of about 100 words which were built performing only sentence and word deletion.

Although the purpose of this corpus is to multi-document summarization, following some ideas of Nóbrega and Pardo (2016) is possible to transform this dataset for the sentence compression task.

Considering that each human-made summary is a document composed by several sentences, each sentence is compressed using word deletion from the source documents. In order to match sentence-compression pairs some heuristics were followed:

- A compressed sentence must be smaller or equal length in respect to the original sentence;
- The compression must be a sub-sequence of words from the source sentence;
- The words present in the compression sentence must be in the same order.

After applying these rules, a new sentence-compression dataset were created with 799 sentence-compression pairs.

4.1.2 Data Augmentation

During the training phase, data augmentation was performed on the English dataset. This method prevents the model to overfit by creating new examples, helping the model to generalize better.

For all the training instances where there are two or more not continuous spans of tokens, that were not taken into consideration for the compressed version, it is possible to create a new training instance for each combination of the spans with the tokens in the compression version, respecting the same order from the original sentence.

In Figure 4.1 we show the different combinations for an example sentence. In this example, the set of spans that were not considered for the ground truth compression were:

- *Studies and surveys have found that*
- *differently*

Respecting the original sequence of tokens, there are two possible training instances that can be produced, as illustrated in Figure 4.1

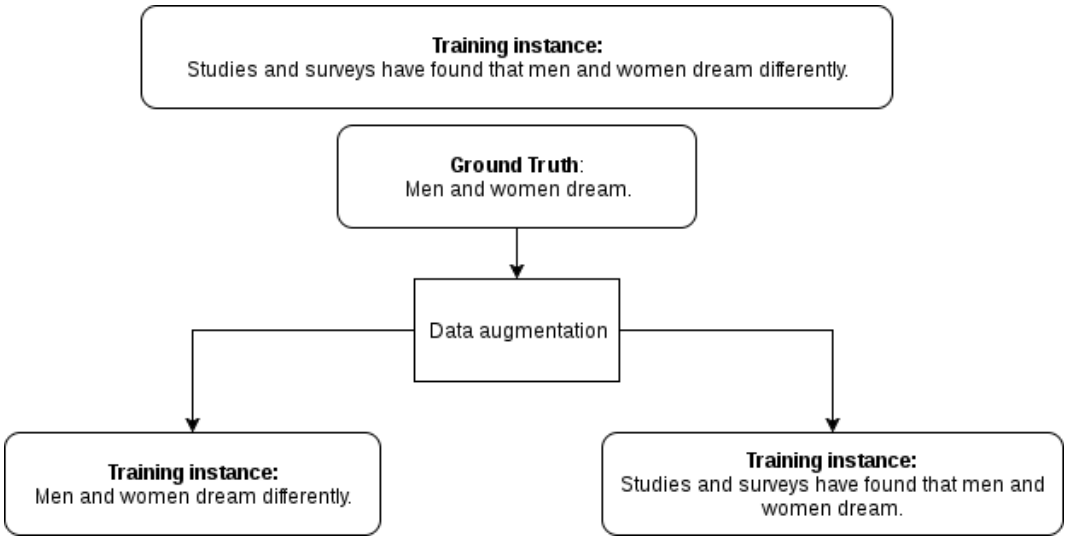

Figure 4.1: Data augmentation process.

4.1.3 Evaluation Metrics

The system outputs a binary sequence of labels with fixed length, which is represented by $y = (y_0, y_1, \ldots, y_n)$, where $y_i = 0$ (e.g., negative class) represents a token that is deleted from the original sequence and $y_i = 1$ (e.g., positive class) indicates that the token remains. Also, it was defined for each input sequence a maximum length of 80 tokens.

In the context of this work, in a sequence the $TruePositive$ are the tokens that the system predicted correctly, $FalsePositive$ are the words that remained in the compression but should have been deleted, and $FalseNegative$ are the tokens that should have not been deleted from the source sentence. Last, $TrueNegative$ are the tokens that the system deleted correctly.

The precision, in Equation 4.1, is the ability of the classifier not to label as positive a sample that is negative.

$$Precision = \frac{\sum TruePositive}{\sum(TruePositive + FalsePositive)} \tag{4.1}$$

The recall, in Equation 4.2, is the ability of the classifier to find all the positive samples.

$$Recall = \frac{\sum TruePositive}{\sum(TruePositive + FalseNegative)} \tag{4.2}$$

The following metrics were used to measure the effectiveness of the system: word-based F1-score, accuracy, per-sentence accuracy, and compression rate.

Word-based F1-score computes the recall and precision in terms of tokens kept in the golden

compression and the generated compression.

$$\text{Word-based F1} = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)} \tag{4.3}$$

Accuracy is defined as the percentage of total tokens correct in the compression.

$$\text{Accuracy} = \frac{\sum(TruePositive + TrueNegative)}{\sum(TruePositive + TrueNegative + FalsePositive + FalseNegative)} \tag{4.4}$$

Finally, per-sentence accuracy represents the percentage of compressions that the model fully reproduced from the test set, and compression rate represents the percentage of shrinkage from the source sentence to the compression (the number of characters in the compression divided by original sentence).

4.1.4 Experimental Settings

In this work, the architecture was implemented using the framework Keras (Chollet et al., 2015) and the parser of spaCy (Honnibal and Johnson, 2015) library.

The word embeddings used were the 300-dimension fastText pre-trained embeddings (Mikolov et al., 2018). If a token does not appear in the vocabulary an embedding is generated based on the character n-grams of the word. The numbers were replaced with a tag NUM and associated with the corresponding tag embedding.

The POS and dependency embeddings have a 10-dimensional and a 40-dimensional vector, respectively, and their weights are updated during training. The main architecture has a two stacked BiLSTM interleaved with a dropout (Srivastava et al., 2014) layer whose value is 0.2. We defined the value 80 as maximum for an input sequence.

The model was trained, with early stopping, using Adam (Kingma and Ba, 2015) as optimizer, with a learning rate initialized as 0.001, and a batch size of 32. The dimension of the hidden-layers of BiLSTM are 200.

The majority of the parameters above were selected based on the work of Reimers and Gurevych (2017), which describes the best parameters to achieve better performance on sequence labeling tasks.

Before evaluating the model Portuguese it was re-trained using a strategy of 5-fold cross validation with the same architecture and parameters described above.

4.2 Results

Section 4.2.1 presents the results achieved on the English dataset compared to state-of-art methods on the same task. Moreover, in Section 4.2.2 describes the results achieved on the Portuguese dataset using the previous model trained on the English dataset. Finally, in section 4.3 a discussion about the results is provided.

4.2.1 Automatic Evaluation on English

The system was evaluated by taking the first 1000 sentences pairs from the test set, following the same practise as Filippova et al. (2015) and Wang et al. (2017). In this work, we did not perform manual evaluation on the sentences predicted.

We compare our approaches with the following baselines that were evaluated also on the first 1000 sentences-compression pairs from the Google News data set:

- LSTM: This is the basic model of Filippova et al. (2015) which have used a sequence to sequence paradigm. In short, the architecture of the network is based on three stacked LSTM layers with a softmax output layer;
- LSTM+: Filippova et al. (2015) proposed an advanced version of their model which uses the same architecture but the input concatenates the current word embedding, parent word embedding of the current word in the dependency tree, and three bits indicating if the parent word has been seen in the compression;
- BILSTM: In this setting, Wang et al. (2017) uses as base model an architecture of three-layered BiLSTM with a softmax as the last input layer;
- BiLSTM+SynFeat: Wang et al. (2017) also proposes an advanced version where they incorporate syntactic features into their model. In the input layer, they combine word, POS and dependency embeddings into a single vector.

Although previous work uses the same dataset, there is a huge difference in size. While Filippova et al. (2015) used a dataset of about 2,000,000 sentence pairs to train their models, Wang

et al. (2017) trained with only 8,000 samples. As described above the dataset used in this work to train the models in English consists in about 200,000 sentence-compression pairs.

Before the incorporation of the CRF layer, it was performed the training of an architecture using LSTM and another one using BiLSTM using only word embeddings as input.

There were considered different syntactic features to incorporate in the model. Different combinations between these features were also tested in the input of the model:

- Word + POS embeddings;
- Word + POS + Dependency relation embeddings;
- Word + POS + Parent word + POS parent embeddings;
- Word + POS + Parent word + POS parent + Dependency relation embeddings.

The set of features that achieved the best results was composed by word, POS, and dependency relation embeddings, which resulted in the following model, BILSTM+CRF+SynFeat.

The results of the automatic evaluation on the Google News dataset are reported in Table 4.1.

	Word F1	Per-sentence Accuracy	Accuracy	Compression Ratio
LSTM (Filippova et al. (2015))	0.8	0.3	-	0.39
LSTM+PAR+PRES (Filippova et al. (2015))	0.82	0.34	-	0.38
BILSTM (Wang et al. (2017))	0.75	-	0.76	0.43
BiLSTM+SynFeat (Wang et al. (2017))	0.8	-	0.82	0.43
LSTM+Softmax	0.79	0.16	0.83	0.41
BiLSTM+Softmax	0.83	0.25	0.86	0.39
BiLSTM+CRF	0.83	0.29	0.86	0.40
BILSTM+CRF+SynFeat	0.84	0.31	0.87	0.41

Table 4.1: Automatic evaluation of the systems on the Google News data set.

4.2.2 Automatic Evaluation on Portuguese

The previous models were also evaluated on the Portuguese dataset, which contains 799 sentence-compression pairs. Due to the size of the data, the re-training of the model in this language uses a 5-fold cross validation. As explained in Section 3.3, the Portuguese and English embeddings were projected into a shared space, making it possible to use in Portuguese the previously trained models on the English dataset. The results are reported in Table 4.2.

	Word F1	Per-Sentence Accuracy	Accuracy	Compression Ratio
BiLSTM+Softmax	0.73	0.07	0.73	0.57
BiLSTM+CRF	0.75	0.19	0.74	0.57
BiLSTM+CRF+SynFeat	0.7	0.11	0.7	0.57

Table 4.2: Automatic evaluation of the models on the Portuguese dataset.

4.3 Discussion

In this work, when evaluating the proposed architecture on English, even the model with no syntactic features outperforms the current neural deletion sentence compression systems in terms of word-based f1 score and accuracy.

Although the incorporation of syntactic features could lead to a better performance of the model by capturing grammatical relations, the results are not significant better. Some compressions predicted by our model can be seen at Table 4.3.

Besides the majority of compressions seems to be grammatically correct, there are some examples which the model decides to remove all the words from the source sentence. This might happen because there is not any constraint to ensure the minimum size of a sentence.

Although the word-based f1 score is better, the per-sentence accuracy metric was not better or equal: we think this is due to the size of the training dataset. While Filippova et al. (2015) trained with about two million sentence-compression pairs, the training of our models were made with two hundred thousand instances, leading to an inferior result on this metric.

Since the dataset used in Portuguese was provided by this work, there are not previous models which we can compare. However, it is possible to verify that the models with a CRF layers benefit when fully reproducing the compressions on the test set.

It is interesting to verify that the model which performs better is the one without any syntactic features. Although previous work reports in most cases better performances using syntactic features, here we can verify that across models there is no benefit in adding these type of features. One of the reasons could be the size of training data, which did not allow the model to capture some important grammatical rules when compressing.

Although we tried to improve the results with data augmentation and using an auxiliary loss during the training, the results achieved were not good enough to consider.

The results achieved demonstrated that this task benefits from the use of the CRF classifier

as last the layer. The ability to decode the best global sequence of labels taking into account the correlations between labels allows to output a better compression on both languages.

S: In response to a question from NDP Treasury Board critic, Mathieu Ravignat on Tuesday, Clement told the House of Commons Tuesday that contracting out government services reduces costs.
G: Clement told the House of Commons Tuesday that contracting out government services reduces costs.
P: Contracting out government services reduces costs.
S: Floyd Mayweather is open to fighting Amir Khan in the future, despite snubbing the Bolton-born boxer in favour of a May bout with Argentine Marcos Maidana, according to promoters Golden Boy."
G: Floyd Mayweather is open to fighting Amir Khan in the future.
P: Floyd Mayweather is open to fighting Amir Khan.
S: Studies and surveys have found that men and women dream differently.
G: Men and women dream.
P: Men and women dream differently.
S: Interethnic relations are not a field for political games, Kazakhstan's President Nursultan Nazarbayev declared in his speech at the Assembly of People of Kazakhstan, Tengrinews reports.
G: Interethnic relations are not a field for political games.
P: Interethnic relations are not a field for political games.

Table 4.3: Sentences and compressions from the Google News data set. S: Input sentence. G: Ground truth compressed sentences. P: Compressed senteces predicted by BILSTM+CRF+SynFeat

5 Conclusions and Future Work

In this work, it was proposed a different neural architecture for sentence compression and the extension of this task into a cross-lingual setting.

It was demonstrated that the current neural deletion sentence compression systems benefit from the use a CRF classifier in the last layer. The proposed architecture achieve better or close results comparing to the current neural deletion approaches. Although the incorporation of syntactic features improved the architecture, the results are not significant better than the base model with only word embeddings.

Across languages the proposed architecture also benefits, however it is interesting to verify the model with syntactic features have a negative impact on the Portuguese results. This could be due to the size of the dataset which did not enable the model to learn enough syntactic information.

In the future, it would be interesting to modify the method proposed by Filippova and Altun (2013) to build a larger corpus of sentence-compression pairs in Portuguese. Furthermore, the use of a larger corpus for the English language may increase the performance of the per-sentence accuracy metric, generating more readable and comprehensible compressions.

One way to improve the prediction of the neural compression system, and avoid the problem of sentences with all the words removed would be the use of a different auxiliary loss function which takes into account the size of each sentence. This function could help the grammaticality of a compression.

Bibliography

Almeida, M., Almeida, M. S., Martins, A., Figueira, H., Mendes, P., and Pinto, C. (2014). Priberam Compressive Summarization Corpus: A New Multi-Document Summarization Corpus for European Portuguese. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 146–152.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Clarke, J. and Lapata, M. (2008). Global Inference for Sentence Compression: An Integer Linear Programming Approach. *Journal of Artificial Intelligence Research*, 31:399–429.

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

Crammer, K. and Singer, Y. (2003). Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3.

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, L., and Vinyals, O. (2015). Sentence Compression by Deletion with LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Filippova, K. and Altun, Y. (2013). Overcoming the Lack of Parallel Data in Sentence Compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491.

Filippova, K. and Strube, M. (2008). Dependency Tree Based Sentence Compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32. Association for Computational Linguistics.

Forney, G. D. (1973). The Viterbi Algorithm. *Proceedings of the Institute of Electrical and Electronics Engineers*, 61(3):268–278.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8).

Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for Sequence Tagging. *arXiv preprint arXiv:1508.01991*.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. *Procedings of the International Conference on Learning Representations*.

Ma, X. and Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computacional Linguistics*, pages 1064–1074.

Martins, A. F. and Smith, N. A. (2009). Summarization with a Joint Model for Sentence Extraction and Compression. In *Proceedings of the Association for Computational Linguistics Workshop on Integer Linear Programming for Natural Language Processing*.

McDonald, R. (2006). Discriminative Sentence Compression with Soft Syntactic Evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Procedings of the International Conference on Learning Representations*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Niklaus, C., Bermeitinger, B., Handschuh, S., and Freitas, A. (2017). A Sentence Simplification System for Improving Relation Extraction. *arXiv preprint arXiv:1703.09013*.

Nóbrega, F. A. A. and Pardo, T. A. S. (2016). Investigating Machine Learning Approaches for Sentence Compression in Different Application Contexts for Portuguese. In *International Conference on Computational Processing of the Portuguese Language*, pages 245–250. Springer.

Plank, B., Søgaard, A., and Goldberg, Y. (2016). Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Annual Conference of the Association for Computational Linguistics*, pages 412–418.

Reimers, N. and Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*.

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

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstruture of Cognition, MIT Press*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wang, L., Jiang, J., Chieu, H. L., Ong, C. H., Song, D., and Liao, L. (2017). Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.