Sentence Compression on Domains with Restricted Labeled Data Availability

Felipe Melo Soares, Ticiana L. Coelho da Silva, and Jose F. de Macêdo
Insight Data Science Lab, Fortaleza, Brazil

Keywords: Sentence Compression, Text Summarization, Natural Language Processing.

Abstract: The majority amount of information available on the Web remains unstructured, i.e., text documents from articles, news, blog posts, product reviews, forums discussions, among others. Given the huge amount of textual content continuously produced on the Web, it has been challenging for users to read and consume every document. Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document. Sentence compression can improve text summarization by removing redundant information, preserving the grammaticality and the important content of the original sentences. In this paper, we propose a sentence compression neural network model that achieved promising results compared to other neural network-based models, even when trained with smaller amounts of data. Rather than training the model only with the words from the training set, the proposed model was trained with different features extracted from the texts. This improves the ability of the model to decide whether or not to retain each word in the compressed sentence.

1 INTRODUCTION

With the internet popularization, there has been a massive growth in the amount of unstructured data available to users. The majority of data is textual, i.e., documents as articles, news, blog posts, product reviews, forums discussions, among others. Given the vast amount of textual content produced continuously on the Web, it has been a challenge for users to read and consume all documents. Thus, text summarization systems play an essential role in this domain.

Automatic text summarization systems often produce summaries by extracting the most relevant sentences from the original documents. Usually, they rank the sentences by rating each one and then select the top-ranked to create the summary, trying to ensure the maximum amount of information with minimum redundancy (Gupta and Lehal, 2010; Nallapati et al., 2016). However, the sentences of the original texts could not be appropriate to compose a summary, since when previously written, they do not necessarily have the same size and conciseness constraints that a summary usually has (Finegan-Dollak and Radev, 2016).

Sentence compression can improve text summarization by removing redundant information, preserving the grammaticality, and the important content of the original sentences (Jing, 2000; Knight and Marcu, 2000; Rush et al., 2015). Many methods to automatically perform sentence compression have been proposed over the years (Jing, 2000; Knight and Marcu, 2000; Knight and Marcu, 2002; McDonald, 2006; Clarke and Lapata, 2008; Cohn and Lapata, 2008; Filippova and Altun, 2013). With the recent growth of computational power availability and the popularization of neural networks, methods that use different variations of neural networks have pushed the state-of-the-art of sentence compression one step further (Filippova et al., 2015; Rush et al., 2015; Chopra et al., 2016; Wang et al., 2017; Févry and Phang, 2018). However, these models still present some limitations.

First, a large amount of labeled data, often millions of sentences, is required to train these models (Filippova et al., 2015; Rush et al., 2015; Chopra et al., 2016), whereas acquiring this amount of data may be an arduous or even impossible task for some text domains or languages. Moreover, when one of these models is trained in a specific domain, it tends to absorb certain aspects of it that ultimately prevent it from achieving the same performance if used with sentences from other domains (Wang et al., 2017).

This paper aims to present a novel neural network-based model capable of generalizing the sentence compression task even when trained with smaller

Soares, F., Coelho da Silva, T. and F. de Macêdo, J.
Sentence Compression on Domains with Restricted Labeled Data Availability.
DOI: 10.5220/0008958301300140
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 130-140
ISBN: 978-989-758-395-7; ISSN: 2184-433X

amounts of data (thousands of sentences instead of millions).

Also, we developed a sentence processing pipeline to minimize the occurrence of rare words that have few contributions to the model's learning by replacing them by common words. This maximizes the occurrence of similar sentences and, consequently, improves the model's ability to extract relevant information from the data. Moreover, in our experiments, we also show that approaches like Sumy (Sumy, 2015) (or Alyien (Aylien, 2011), Tools4Noobs (Tools4Noobs,), among others), that is a widely used platform for automatic text summarization can not compress short sentences as our approach.

The paper is organized as follows: Section 2 formally defines the problem of sentence compression. Section 3 presents some details of the sentence processing pipeline used to train our proposed model. Section 4 presents the proposed model novel architecture to perform sentence compression by word removal. Section 5 presents some related works. Section 6 describes the experiments and discusses their results. Finally, Section 7 draws the conclusion and future works.

2 PROBLEM DEFINITION

Let $\mathcal{V}$ be the word vocabulary and $\mathcal{W} = (w_0, \ldots, w_n)$ a sentence composed by a sequence of words $w_i \in \mathcal{V}$, $\forall i \in \{0, \ldots, n\}$. We aim to compress $\mathcal{W}$ in order to generate a new sentence $\mathcal{Y} = (y_0, \ldots, y_m)$ such that $n \geq m$ and $\forall j \in \{0, \ldots, m\}, \exists i \in \{0, \ldots, n\}$, such that $y_j = w_i$ and $j \leq i$, i.e., $\mathcal{Y}$ is a compression of $\mathcal{W}$. Moreover, $\mathcal{Y}$ must retain as much information as possible from $\mathcal{W}$ and must be grammatically correct. It is possible to interpret this problem as a sequence labeling problem in which we aim at training a model $\mathcal{M}$ such that $\forall w_i \in \mathcal{W}$, $\mathcal{M}(w_i)$ is 1, if $w_i \in \mathcal{Y}$, i.e., if the word w_i belongs to the compressed sentence, and 0 otherwise.

3 SENTENCES PROCESSING PIPELINE

Neural network-based sentence compression models commonly rely on large amounts of labeled data to be trained. The reason is textual information entails a high burden of dependencies and relationships that are often difficult to be learned by these models if no information beyond the words is provided. Typically, these models try to learn the co-occurrence of words and phrases to attempt to extract some meaningful information. However, if the neural network models are trained with a small textual data set, these models may end up learning very well the relationship between the words that occur in that data. But they are unable to infer anything about words that are out of the training set, which makes them biased.

As an attempt to minimize the impact of a small set of labeled data that some domains may have, we propose a sentence processing pipeline to decrease the influence of words with few occurrences by replacing them by words of similar meaning. For instance, the words *John* and *Mary* are both names of persons, and they can be simply replaced by the word *name*. So instead of two words that occur rarely in the training set, we sum up the occurrence of both into a single word. This processing provides more examples with similar structures to be used in the training phase. Note that these replacements do not change the order or the number of words in a sentence. Figure 1 illustrates the pipeline steps described below, and they are briefly discussed as follows.

- Contraction Removal: The first step consists of removing possible contractions that occur in sentences. Contracted expressions like *don't* are replaced by their non-contracted version (*do not*). This step is important to ensure that all words are explicitly described for future steps.
- Tokenization: is an essential step of any natural language processing task that operates at the word level. In this step, the sentences are segmented into tokens, which usually represent each word of a text. For instance, after tokenization the sentence *Attorney General Eric Holder has been hospitalized*, the list of tokens is: (*Attorney*, *General*, *Eric*, *Holder*, *has*, *been*, *hospitalized*).
- Part-of-Speech (POS) Tagging: To extract grammatical information from the tokens, we perform a part-of-speech tagging step. Part-of-speech tagging is a common task in natural language processing, and it identifies the part-of-speech tags of each token within the sentence. The outputted tags of each sentence are later used as input to our proposed sentence compression model.
- Dependency Parse: To comprehend how words or tokens within the sentence are related to each other, our approach does the dependency parse step. At this step, for each word or token, our approach extracts: 1) the parent word, 2) the dependency relationship with its parent, and 3) the number of nodes on the dependency subtrees. Figure 2 shows an example of a syntactic dependency tree

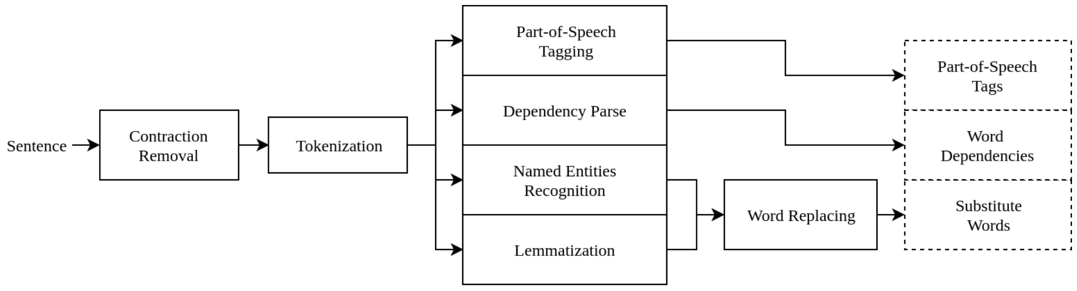


Figure 1: Sentence processing pipeline steps.

for a sentence with POS tags for each word. For instance, in the sentence *Attorney General Eric Holder has been hospitalized*, the word *Holder* is the parent of the word *hospitalized*. The word *hospitalized* has a relation with *Holder* named as *nsubjpass* which means that *Holder* is a nominal subject passive relative to *hospitalized*. Beside that, the number of nodes on the dependency subtree of *hospitalized* is one.

- Named Entities Recognition (NER): Named entities are frequently found in any text, and they tend to be words that present low frequency of occurrence. Commonly, the text vocabulary becomes large with many words that rarely occur (no more than two or three times), and thus it's difficult for the model to learn how these words are related. To avoid this problem, named entities are identified and later replaced by common and semantic values. For instance, the names of buildings, roads, countries, and cities are essentially locations. Consequently, they are all replaced by the word "location". The same happens for person names, objects, organizations, groups, numbers, etc. To a more detailed inspection of substitutions, the Table 1 maps each used named entity type with its substitute word.
- Lemmatization: The lemmatization step identifies the lemma of each word or token. The lemma of a word is its base, canonical or dictionary form, usually in nominative singular for adjectives and nouns and infinitive, without *to*, for verbs. For instance, the lemma of the words *play*, *played* or *playing* is simply the word *play*. As in named entities recognition step, the lemma of each word is later used in replace of the word to reduce the occurrence of words that have the same semantics and different spellings.
- Word Replacing: This step is responsible for decreasing the occurrence of rare words. As a word is identified as a named entity, it is immediately replaced by another word that carries part of its named entity type semantics, as indicated in Table 1. While the words' lemmas simply replace the words not identified as named entities.

At the end of the sentence processing pipeline, the sentence "Attorney General Eric Holder has been hospitalized", is written as the following sequence of tokens: (*attorney*, *general*, *name*, *name*, *have*, *be*, *hospitalize*). Note that, although that sequence of words does not produce a grammatical sentence, it provides to the model a better understanding of relationships between the words.

4 MODEL ARCHITECTURE

In this section, we explain the proposed model architecture to perform sentence compression.

4.1 Feature Selection

Since we extracted some important word features during the sentence processing pipeline (explained in the previous section), now we need to present their representations as inputs to our model. This section explains which features were used, the reasons why we chose to use them, and their representations as input to the model. We sought three main goals while determining which features to use. First, to retain original information of each word. Second, to obtain a structural representation of the sentence that shows the importance of each word and the role it plays in the sentence. And finally, to show the dependencies between words explicit, since the information that a word has been retained or removed in the compressed sentence may be important for deciding the next word. Thus, three feature groups were used. Following, we list all used features and describe their representations:

- Semantic Feature: The word itself is the main input feature of the model. It contains all relevant information that has not been manually extracted from other features, and, of course, it is still useful for classifying whether or not it should be retained

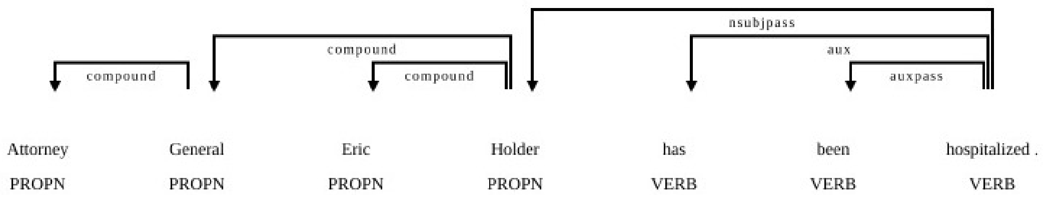


Figure 2: Example of dependency tree with the part-of-speech tag for each word.

Table 1: Table with the replacement words for each named entity type.

Named Entity Type	Description	Substitute Word
PERSON	Person	name
NORP	Nationalities or religions or political groups	group
FAC	Buildings, airports, roads, bridges, etc.	location
ORG	Companies, agencies, institutions, etc.	organization
GPE	Countries, cities, states, etc.	location
LOC	Non-GPE Locations	location
PRODUCT	Objects, vehicles, food, etc.	object
EVENT	Names of hurricanes, battles, wars, etc.	name
WORK_OF_ART	Book titles, songs, etc.	name
LAW	Named documents made by law.	law
LANGUAGE	Name of any language.	language
DATE	Absolute or relative dates or periods.	date
TIME	Moments less than a day.	moment
PERCENT	Percentage, including "%".	number
MONEY	Currency values including unit.	number
QUANTITY	Measures such as distance or weight.	measurement
ORDINAL	Ordinal numbers.	number
CARDINAL	Numerals that do not fall into any another type	number

in the compressed sentence. It is represented by a pre-trained embedding vector using some word embedding model like GloVe (Pennington et al., 2014) or Skipgram (Mikolov et al., 2013);

- Structural Features: During sentence preprocessing, we perform a Part-of-Speech (POS) tagger over the sentences. The POS tags are used as a feature to characterize a word structurally. Embedding vector also represents the POS tags, and they are trained throughout the training phase of the model. Moreover, in the preprocessing phase, the sentence dependency parse is performed, and the syntactic dependency tree is constructed, as shown in Figure 2. By using the dependency tree, our approach counts the number of nodes in the subtrees of each word. This amount is used as a feature to our proposed model. It roughly means the importance of each word in the sentence information.

- Dependency Features: We intend to explicitly provide the dependencies between the words in a given sentence. Thus, we use the position of each word and the position of its parent word, both represented by a one-hot vector. We also capture as a feature, the dependency relationship between a word and its parent which is represented as an embedding vector trained along with the model.

Given a sentence $\mathcal{W} = (w_0, w_1, ..., w_n)$, let e_i be the embedding vector representing a word w_i, t_i the embedding vector representing the POS tag of w_i, d_i the embedding vector representing the dependency relation between w_i and its parent word, o_i the one-hot vector of i, p_i the one-hot vector of the position of w_i's parent and s_i the total nodes in w_i dependency subtree for $i \leq n$, our model combines all these features, by concatenating them, into a single vector to be used as input:

$$x_i = e_i \oplus t_i \oplus d_i \oplus o_i \oplus p_i \oplus s_i, \qquad (1)$$

where $\oplus$ means the concatenation of vectors.

4.2 Model Architecture

As in (Filippova et al., 2015), the trained model is capable of, given a sentence $W = (w_0, w_1, ..., w_n)$ where each w_i represents a word of the sentence, classify a word w_i in *retained* or *removed* in the compressed sentence. To this end, our proposed model was designed inspired by the encoder-decoder architecture proposed in (Cho et al., 2014) and (Sutskever Google et al., 2014). This architecture has two main modules: an encoder, responsible for learning a vector representation for a sequence of tokens and a decoder, responsible for generating a sequence of tokens from a vector that represents an encoded sequence.

In general, this architecture is used to build models that can learn how to transform a sequence of tokens into another one. Our model gets inspired by encoder-decoder architecture, however its modules do not necessarily have the same roles. In this paper, the encoder is responsible for generating a vector representation of an entire sentence, and then provides more information about the whole sentence as input for the classifier. This is similar to the original encoder purpose. But, instead of a decoder, we have a classifier that receives as input a sequence of words (represented by its features), and the vector that represents the full sentence encoded, as we can see in Figure 3. Both encoder and classifier will be explained in the next sections.

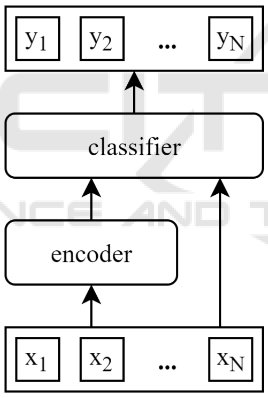


Figure 3: Overview of proposed model.

4.2.1 Sentence Encoder

The encoder was implemented using a bidirectional LSTM stack architecture (Graves et al., 2005). Compared to an ordinary LSTM architecture, bidirectional LSTM has a greater ability to capture contextual information from a sentence, such as dependencies between successive or non-successive words. As you can see in Figure 4, the architecture has two stacks of LSTM networks, interleaved with dropout layers, which help the model to minimize overfitting (Srivastava et al., 2014). The forward stack encodes a sentence $X = (x_0, x_1, ..., x_N)$ in a vector h_e^F. While the backward encodes the same sentence, in reverse order, in h_e^B vector. The vectors h_e^F and h_e^B are then concatenated to compose a new h_e vector that represents the vector in both directions:

$$h_e^F = Forward(X) \tag{2}$$

$$h_e^B = Encoder(X) \tag{3}$$

$$h_e = h_e^F \oplus h_e^B \tag{4}$$

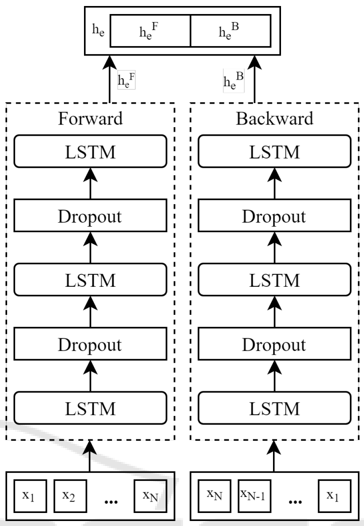


Figure 4: Overview of sentence encoder.

4.2.2 Classifier

Our classifier is similar to the model proposed in (Wang et al., 2017). The main difference is in the input data. Rather than receiving only word-related attributes, the proposed model also receives a vector representing the entire sentence concatenated to each word input. Thus, it is expected that at the moment of each word classification, the model has more information as input than just the word itself, improving its ability to decide whether or not to retain that word in the compressed sentence:

$$h_i^F = Forward(h_e \oplus x_i) \tag{5}$$

$$h_i^B = Backward(h_e \oplus x_i) \tag{6}$$

$$h_i = h_i^F \oplus h_i^B \tag{7}$$

$$y_i = h_i \cdot W + b \tag{8}$$

Figure 5 illustrates this architecture in details along a sequence of predictions.

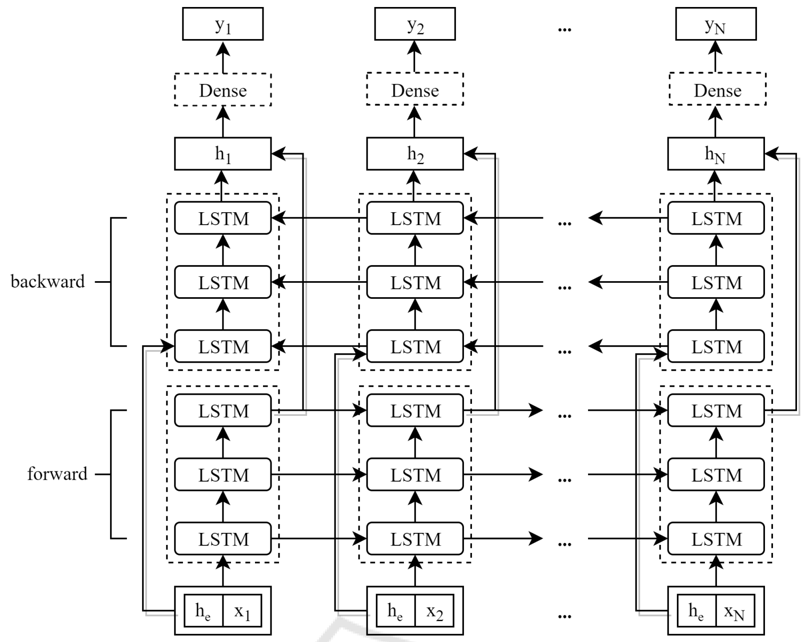


Figure 5: Detailed classifier unrolled along a sentence. The dropout layers were intentionally omitted for readability.

5 RELATED WORKS

In the beginning, sentence compression aimed to improve the quality of summaries generated by extractive text summarization systems. However, different names and definitions can be associated with text summarization. In the literature, we find this task with three different names: sentence compression (Knight and Marcu, 2000; McDonald, 2006; Cohn and Lapata, 2008; Filippova et al., 2015), sentence summarization (Rush et al., 2015; Chopra et al., 2016) and sentence reduction (Jing, 2000). Sentence reduction refers to the task of removing extraneous phrases from a sentence (Jing, 2000), while sentence compression and sentence summarization both can be defined as the task of, given a sentence, obtain a shorter, but still informative and grammatical sentence. However, it is common to find in the literature some distinction between these two. Sentence compression is often used as synonymous of sentence reduction, i.e., describe methods that focus on removing unessential expressions from the sentences only. The term sentence summarization is used to describe methods that seek to reduce the length of sentences, i.e., generating sentences from scratch based on the original.

For homogenization purposes, the task of obtaining a shorter sentence from another sentence will be termed as sentence compression, once that is the most frequently used term in the literature.

The work in (Jing, 2000) is the first one that tried to develop a method to compress sentences automatically. (Jing, 2000) focused on removing extraneous phrases of the sentences. So, (Jing, 2000) proposed a system that used multiple sources of knowledge, contextual information, and statistics calculated over a human-written corpus to decide which phrases should be removed from the original sentence. (Knight and Marcu, 2000) proposes two different models to sentence compression: a probabilistic noisy-channel and a decision-tree based model. In the experimentation study, (Jing, 2000) was more conservative, while (Knight and Marcu, 2000) was more aggressive in its compression, retaining fewer important words than the noisy-channel model. In (McDonald, 2006), a large margin-learning algorithm is proposed to model the task as attempting to classify if two words are adjacent or not in the compressed sentence using a large number of features extracted from the words. The work of (Clarke and Lapata, 2008) models sentence compression as an integer linear programming (ILP) problem whose goal is to seek and find an optimal compression given some constraints over the language.

(Cohn and Lapata, 2008) is the first work to explore other rewrite operations, besides deletion, to perform sentence compression. It analyzed the proportion of each common rewrite operation on human-made sentence compression. It found out that word substitution and sentence reordering were also common operations on compressions made by humans.

Given the small amount of parallel data corpus, i.e., corpus with pairs of sentences and its compressions, to train neural networks-based models, (Filippova and Altun, 2013) developed a method for automatically generating a corpus by matching the compressed sentence dependency tree to a sub-tree of the original sentence dependency tree. (Filippova et al., 2015) used the method proposed in (Filippova and Altun, 2013) to create a corpus of millions of pairs of sentences to make feasible the training of a neural network-based model inspired by the advancements in neural machine translation (NMT). The model of (Filippova et al., 2015) implemented a stack of Long Short-Term Memory (LSTM) networks to classify each word in a sentence as retained or removed and evaluated the impact of some syntactic features over the model performance.

After (Cohn and Lapata, 2008), (Rush et al., 2015) was the first work to make a new attempt on doing sentence compression in a non-deletion based approach. Inspired by the attention model of (Bahdanau et al., 2014), widely used in NMT, (Rush et al., 2015) is also the first that named the task as sentence summarization. It proposes an attention-based encoder to learn a vectorial representation for a sentence and uses a neural language model to decode this sentence encoded vector into a new sentence, shorter but with the same meaning of the original. (Chopra et al., 2016) extends the model from (Rush et al., 2015) and replaces the neural language model by a recurrent neural network to create the compressed sentence.

The works of (Jing, 2000), (Knight and Marcu, 2000) and (McDonald, 2006), as first supervised methods, had a concern about the quality of their models when applied to sentences outside the domains they were trained on. However, with the advancement of neural networks-based models, this concern was gradually being neglected. Since so, (Wang et al., 2017) work is the first to investigate how to ensure that a model trained on some domain can generalize the sentence compression task properly into domains under which it has not been trained. To do so, it extends the (Filippova et al., 2015) model using a bidirectional LSTM instead of an ordinary LSTM to capture contextual information better and using a set of specific syntactic features instead of merely using words as performed in (Filippova et al., 2015). Additionally, the work of (Wang et al., 2017), inspired by the work of (Clarke and Lapata, 2008), uses integer linear programming to find the optimal combination of labels.

The problem of sentence compression has been addressed differently over the years. So, it is hard to compare the proposed models and algorithms. The advancement of neural networks brought significant improvement for the field, but also brought the already known problems of data acquisition. Except for the model of (Wang et al., 2017), all presented neural network-based models require large amounts of labeled data to achieve good performances. These models take advantage of these massive amounts of data to extract important features from the words, by themselves, without using any syntactic information. Nevertheless, these massive amounts of data are not always open and publicly available. So, with small amounts of data, these models can not extract, by themselves, the features they need to achieve better performances.

We propose a model that aims to overcome this problem by previously extracting some important features and applying some pre-processing steps. Our model is an extension of (Wang et al., 2017). So, the model proposed in (Wang et al., 2017) is our baseline.

6 EXPERIMENTS

In this section, we discuss the experimental evaluation.

6.1 Dataset

The dataset used in these experiments is a set of 10,000 pairs of sentences (original and compressed) publicly released in (Filippova et al., 2015), both for training and for model evaluation. These sentences were automatically extracted from Google News using a method developed in (Filippova and Altun, 2013). The 10,000 sentences were split into a training set consisting of approximately 8,000 sentences, around 1,000 sentences for the validation set, and finally, the test set with 1,000 sentences. We study how our model and the baseline perform with a small training set. So, we consider three different samples of training set: one with the full 8,000 sentences, another with the first 5,000 sentences, and the last one with the first 2,000 sentences.

6.2 Experimental Setup

In the experiments, we trained all models using Adam (Kingma and Ba, 2014) as the optimizer with the learning rate starting at 0.001 and using cross-entropy as the loss function. The hidden layers of the LSTM for all models were set to 100. The word embedding vectors were set to 100. They were initialized using pre-trained GloVe vectors of the same dimension and were not updated during training. The embedding vectors of part-of-speech tags and dependency relations were both set to 40 dimensions and, unlike the embedding vectors of the words, were updated during the model training. The dropout layers that interleaved the LSTM layers were set to drop 50% of neurons during weight update steps randomly, and finally, all models were trained with a batch size of 30 for 20 epochs each.

We analyzed four variations of sentence compression neural network-based model architectures. Table 2 summarizes the relationship between models and their characteristics described below:

BiLSTM: is the baseline. This model is the one proposed in (Wang et al., 2017). It uses a bidirectional LSTM stack and the combination of embedding vectors of each word, its part-of-speech tag, and its dependency relationship with the word's parent as input features. As in its original work, this model does not use the proposed sentence processing pipeline.

EncBiLSTM: This architecture uses an encoding module to get a vector representation of the full sentence and combines that representation with the same input features of the BiLSTM. The classifier, as in BiLSTM, is a bidirectional LSTM stack, and the model does not use the proposed sentence processing pipeline.

BiLSTM+: This configuration uses the same architecture as the BiLSTM, without using an encoder to get the vector representation of the sentence. However, in addition to the BiLSTM input features, BiLSTM+ uses the word position in the sentence, the word parent position, and the number of nodes in each word's dependency subtree. Besides that, the configuration uses the proposed sentence processing pipeline.

EncBiLSTM+: Finally, EncBiLSTM+ uses the same architecture as the EncBiLSTM. However, in addition to its input features, EncBiLSTM+ also uses the word position in the sentence, the position of the word parent and the number of nodes in the dependency subtree of each word (as BiLSTM+). Moreover, this configuration also uses the proposed sentence processing pipeline.

6.3 Evaluation Metrics

The test set consists of 1,000 pairs of sentences previously picked up. To evaluate the models, we calculated word-level accuracy, F1 score, and the average compression rate of the sentences from the test set against the ground truth. The word-level accuracy measures how many words the model classified correctly. The F1 score is a metric derived from the other two metrics, precision and recall. Precision measures how many retained words were correctly classified and recall measures how many words that must be retained were classified as so. The average compression rate is the average between the compression rate, i.e., the ratio of retained words over the total words count of the sentence, of all sentences from the test set.

In order to compare the quality of compression, we used BLEU (Papineni et al., 2001) and ROUGE (Lin, 2004) for automatic evaluation. BLEU is a metric originally proposed for neural machine translation that claims to be highly correlated with human assessment. ROUGE is a metric widely used in literature to evaluate the quality of text summarization by overlapping parts of the generated compression or summary against a reference or set of references. The most common variations of ROUGE are for overlapping unigrams, bigrams, and longer common subsequence, respectively ROUGE-1, ROUGE-2, and ROUGE-L.

6.4 Experimental Result

The results reported in Table 3 refer to the experiment using 8,000 sentences in the training set, and Figure 6 illustrates the comparative performance of the models, in terms of accuracy and F1 score, with different amounts of training data.

Given a large amount of data (8,000 sentences), the EncBiLSTM+ model performs slightly better than the others, as can be seen in Table 3. F1-score and accuracy show that EncBiLSTM+ performs better than its competitors by predicting whether each word should be retained or removed from the sentences. ROUGE values, especially ROUGE-1, were significantly higher for all models, and slightly higher for EncBiLSTM+ since compressions are naturally subsets of original sentences. As ROUGE measures the overlapping n-grams between compressed and original sentences, so it is expected that ROUGE scores would be high.

The advantage of the EncBiLSTM+ model is probably because the model has two modules, a sentence encoder, and a word classifier. This makes it more robust, but also requires a more substantial amount of data to train. Combining this with the processing pipeline, which somehow facilitates the model training, we understand why the models that used our proposed pipeline performed better.

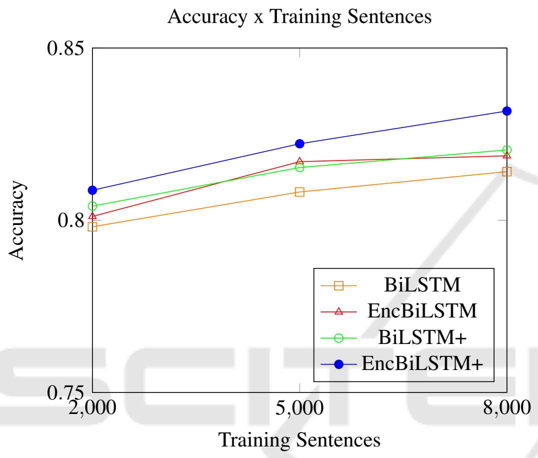
When the amount of training data decreases, the performance of the proposed model drops as well since its architecture has more weights to train than the baseline's for the same amount of data. The BiLSTM+ model uses the same set of features and sentence processing pipeline as the EncBiLSTM+. Remember the difference between them is the encoder phase. For smaller amounts of data the difference be-

Table 2: Relationship between evaluated models and their characteristics.

	Word	Part-of-Speech Tag	Dependency Relation	Encoded Sentence	Word Position	Word Parent Position	Subtree Nodes Count	Sentences Processing Pipeline
BiLSTM	X	X	X					
EncBiLSTM	X	X	X	X				
BiLSTM+	X	X	X		X	X	X	X
EncBiLSTM+	X	X	X	X	X	X	X	X

Table 3: Results from models trained with 8,000 sentences.

	F1	Accuracy	Compression Rate	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
BiLSTM	0.7922	0.8141	0.4211	0.5445	0.7687	0.6330	0.7329
EncBiLSTM	0.7971	0.8187	0.4343	0.5578	0.7748	0.6499	0.7415
BiLSTM+	0.8045	0.8204	0.4451	0.5707	0.7882	0.6672	0.7552
EncBiLSTM+	**0.8149**	**0.8317**	0.4397	**0.5849**	**0.7955**	**0.6802**	**0.7625**



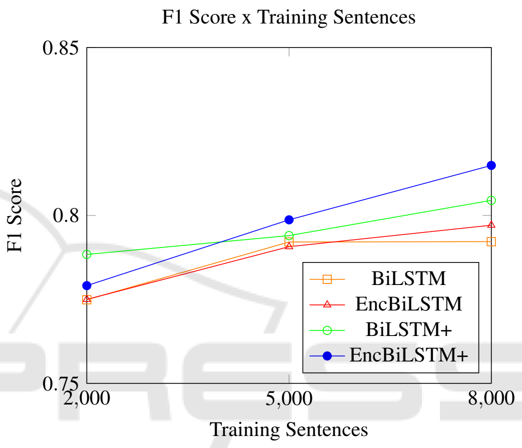


Figure 6: Accuracy and F1 score of models with different amounts of training sentences.

tween the two models decreases considerably to the point that the first has a value of F1 score better than the second for 2,000 training sentences, as you can see in Figure 6.

Notice that BiLSTM+, EncBiLSTM, and EncBiLSTM+ outperform the state-of-the-art model (BiLSTM). This proves that our proposed neural network model and the proposed sentence processing pipeline offer promising results compared to the baseline, even when trained with smaller amounts of data.

6.5 Compressed Sentence Analysis

We show a comparison between EncBiLSTM+ and BiLSTM in Table 4. In the first example, both models outputted quality compression sentences. EncBiLSTM+ generated a compressed sentence shorter than BiLSTM. The second example shows that BiLSTM generated a sentence that is grammatically correct, however, the compressed sentence does not keep the same meaning of the original sentence. While EncBiLSTM+ achieved exactly the same compression of the ground truth. The third example shows an interesting case. EncBiLSTM+ generated a compressed sentence much longer than the ground truth, but more coherent. Since the sentences were automatically extracted, it is possible that some ground truth compressed sentences are not the best compression. Finally, the fourth example shows a case in which both models EncBiLSTM+ and BiLSTM generated quality compressed sentences, yet different, compressions for the original sentence. In Table 4, we also show the compressed sentences outputted using Sumy that is a widely used tool for text summarization. Notice that Sumy can not compress small sentences different from our approach. This is another benefit of our approach.

7 CONCLUSION

We present a more robust neural network-based model to automatically compress sentences by using a handful set of syntactic features and a sen-

Table 4: Comparison between the compressed sentences outputted from the models and the ground truth.

Original:	German Chancellor Angela Merkel early on Wednesday agreed to form a coalition government with the Social Democrats, negotiation sources told AFP, two months after her conservatives won elections but fell short of a full majority.
Ground Truth:	Angela Merkel agreed to form a coalition government with the Social Democrats.
BiLSTM:	Chancellor Angela Merkel agreed to form a coalition government Social Democrats.
EncBiLSTM+:	Angela Merkel agreed to form a coalition government.
Sumy:	German Chancellor Angela Merkel early on Wednesday agreed to form a coalition government with the Social Democrats, negotiation sources told AFP, two months after her conservatives won elections but fell short of a full majority.
Original:	Manufacturing strengthened from China to South Korea last month in a sign that growth risks are abating in Asia and expansion may pick up this quarter.
Ground Truth:	Manufacturing strengthened from China to South Korea.
BiLSTM:	Manufacturing strengthened China.
EncBiLSTM+:	Manufacturing strengthened from China to South Korea.
Sumy:	Manufacturing strengthened from China to South Korea last month in a sign that growth risks are abating in Asia and expansion may pick up this quarter.
Original:	”I am Chelsea Manning, I am a female,” Manning said in a statement released on Thursday, before going on to say that ”given the way I feel and have felt since childhood, I want to begin hormone therapy as soon as possible”.
Ground Truth:	I am Chelsea Manning, I am.
BiLSTM:	I am Chelsea Manning.
EncBiLSTM+:	Chelsea Manning want to begin hormone therapy.
Sumy:	”I am Chelsea Manning, I am a female,” Manning said in a statement released on Thursday, before going on to say that ”given the way I feel and have felt since childhood, I want to begin hormone therapy as soon as possible”.
Original:	Emma Thompson fears young actresses are under ”terrible” pressure to look like skinny fashion models.
Ground Truth:	Emma Thompson fears young actresses are under pressure to look like models.
BiLSTM:	Emma Thompson fears young actresses are under terrible pressure to look like models.
EncBiLSTM+:	Emma Thompson fears actresses are under terrible pressure to look like skinny models.
Sumy:	Emma Thompson fears young actresses are under ”terrible” pressure to look like skinny fashion models.

tence processing strategy to minimize the occurrence of rare words. We experimented with 8,000 sentences as training data, which is considerably less than the number of sentences usually used by neural network-based sentence compression models in the literature. Our results indicate that, although a neural network-based model can perform feature extraction by itself when exposed to a huge amount of training data, it can also benefit from explicitly extracted features to counterbalance the lack of that data. That is the intuition behind the idea of our proposal. We also noticed that our approach performs better to compress small sentences, different from state-of-the-art techniques for summarization. In the future, we expect to evaluate our model against other criteria such as sentence readability, informativeness, and grammaticality as well as try to reduce the number of trainable weights for the model without losing its robustness.

ACKNOWLEDGMENTS

This work is partially supported by the FUNCAP SPU 8789771/2017, and the UFC-FASTEF 31/2019.

REFERENCES

Aylien (2011). Aylien Text Analysis API. https://developer.aylien.com/text-api-demo. [Online; accessed October-2019].

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

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*.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. *Naacl-2016*, pages 93–98.

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

Cohn, T. and Lapata, M. (2008). Sentence Compression Beyond Word Deletion. *Proceedings of COLING*, (August):137–144.

Févry, T. and Phang, J. (2018). Unsupervised Sentence Compression using Denoising Auto-Encoders.

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, L., and Vinyals, O. (2015). Sentence Compression by Deletion with LSTMs. *Emnlp*, lstmsen(September):360–368.

Filippova, K. and Altun, Y. (2013). Overcoming the Lack of Parallel Data in Sentence Compression. *Emnlp*, (October):1481–1491.

Finegan-Dollak, C. and Radev, D. R. (2016). Sentence simplification, compression, and disaggregation for summarization of sophisticated documents. *JASIST*, 67:2437–2453.

Graves, A., Fernández, S., and Schmidhuber, J. (2005). Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer.

Gupta, V. and Lehal, G. S. (2010). A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Jing, H. (2000). Sentence reduction for automatic text summarization. *Proceedings of the sixth conference on Applied natural language processing -*, pages 310–315.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Knight, K. and Marcu, D. (2000). Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Knight, K. and Marcu, D. (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

McDonald, R. T. (2006). Discriminative Sentence Compression with Soft Syntactic Evidence. *Proceedings of EACL*, pages 297–304.

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

Nallapati, R., Zhai, F., and Zhou, B. (2016). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rush, A. M., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization.

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.

Sumy (2015). https://pypi.org/project/sumy/. [Online; accessed October-2019].

Sutskever Google, I., Vinyals Google, O., and Le Google, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Technical report.

Tools4Noobs. Http://www.tools4noobs.com/summarize/. [Online; accessed October-2019].

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. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1385–1393.