UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Neural Paraphrasing Generation System

MASTER'S THESIS

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Author: Jaume Zaragoza Bernabeu

Tutor: Francisco Casacuberta Nolla

Course 2018-2019

Abstract

A paraphrase is a restatement of the meaning of a text or passage using other words. There are many applications of paraphrasing like rewording texts while writing, giving alternative translations for a target sentence, identifying similar sentences, getting synonyms or expanding search queries to match additional information. In order to help all those applications, the aim of the project is to build a system that can provide paraphrases of a given phrase. To build that system, we will explore different state-of-the-art techniques based on neural networks, and more specifically, inspired by neural machine translation recent work. Firstly, we will perform an unsupervised task that focuses in the generation of sentence embeddings (vectors of real numbers) representing semantic information in a continuous space. To generate sentence embeddings we will use large corpora, with millions of sentences of of public available books or subtitles of TV series, films and documentaries. Then, the embeddings will be tested in terms of semantic relatedness (what degree of similarity two sentences have) and paraphrase identification (if two sentences are paraphrases). Finally, we will build a paraphrase generation model using these embeddings to improve its performance.

Key words: Paraphrasing, Sentence Embeddings, Neural Machine Translation

Resum

Entenem com a paràfrasi l'acte de reescriure un text amb paraules diferents mantenint el seu significat. Hi podem trobar moltes aplicacions de la paràfrasi tals com reescriure paraules mentre s'escriu una text, proporcionar traduccions alternatives per a una frase objectiu, identificar frase similars, obtenir sinònims o expandint consultes de cerca per a trobar més informació. Amb l'objectiu d'ajudar a totes aquestes aplicacions, l'objectiu del projecte és construir un sistema que proporcione paràfrasis a partir d'una frase donada. Per a construir aquest sistema, explorarem diferents tècniques de l'estat de l'art basades en xarxes neuronals, més concretament, inspirades en traducció automàtica neuronal. Primerament realitzarem una tasca no supervisada que es centrarà en la generació d'embeddings de frases (vectors de nombres reals) que representen la informació semàntica en un espai continuu. Per a generar aquests embeddings usarem corpus de gran tamany, amb milions de frases de llibres públics o de subtítols de series de televisió, pelůlícules i documentals. Després aquests embeddings seran provats en tasques sobre relació semàntica (quin grau de similitud tenen dues frases) i identificació de paràfrasi (si dues frases són paràfrasi). Finalment, construirem un sistema de generació de paràfrasi usant aquests embeddings per a millorar el seu rendiment.

Paraules clau: Paràfrasi, Representació vectorial de frases, Traducció neuronal

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

1.1 Paraphrasing Definition

For a computer, like other natural language processing tasks, paraphrasing, i.e. identifying if two or more sentences can be replaceable, or to search which sentences are equivalents of one given sentence is still very complicated. Probably one of the reasons that explains the complexity of this task is because not even humans have a unique definition of what paraphrasing is. Since years, there has been a lot of research about paraphrasing not only from the point of view of natural language processing (NLP), but also in linguistics. Researchers define paraphrasing in many different ways including the expressions *"convey the same information"*, *"can be replaced by each other"* or *"semantically equivalent"*.

According to Wikipedia, *"a paraphrase /pærfrez/ is a restatement of the meaning of a text or passage using other words. The term itself is derived via Latin paraphrasis from Greek παράφρασις, meaning "additional manner of expression"*. Furthermore, [Ho+12] indicates that a generalised definition could be: *"paraphrases are different words, phrases or sentences that express the same or almost the same meaning"*. But even having a general definition, when it comes to determine if two sentences, phrases or words are paraphrases, sometimes not even even humans agree. The task becomes, then, very difficult to formalise and to automate for a machine.

When dealing with paraphrasing, one can identify various types of paraphrases:

- Lexical paraphrases: synonyms or words that are replaceable in some similar contexts or carry out the same syntactic role. For example: *secured→guaranteed* or *shows→contains*.
- Phrasal paraphrases: when more than one word is involved but not full sentences. For example: *extended an invitation→be invited to* or *portions of the→some parts of the*.
- Sentential paraphrases: the ones that cover full sentences. For example: *Two people are riding a motorcycle→A bike is being driven by two people*.

Another way to categorise paraphrases is paraphrasability, a form of describing in which degree, or in which way, words or phrases are replaceable or interchangeable. On the one hand, there are unidirectional paraphrases where one carries the meaning of the other but not in the reverse way. On the other hand, bidirectional paraphrases are paraphrases that can be interchangeable in both directions. For example: *cat* can be replaced by *animal*, but *animal* cannot replaced by *cat* as it represents a wider meaning including other animals like dog, bird, etc. An example of bidirectional lexical paraphrase is *common* and *usual*. These are mutually replaceable in many contexts, although not in all of them as we will see below.

Context sensitivity is, then, another important aspect of paraphrases. There are paraphrases that can be replaced when they are used alone (context insensitive) and others that can be replaced only in certain contexts (context sensitive). With the examples above, *common* and *usual* are context sensitive because there are contexts where they are not interchangeable. For example, in the sentence *We all have a common objective*, the word *usual* cannot be used to replace *common* because the meaning of the sentence would change completely.

There are two main tasks involving paraphrasing: paraphrase extraction and paraphrase generation. Paraphrase extraction also involves paraphrase detection, consisting in identifying words or phrases that have the same or similar meaning. Once the phrases have been identified, they are extracted in order to construct paraphrase tables, databases or corpora. Then, these resources are used in the process of paraphrase generation (which is more commonly known as paraphrasing), that is the production of candidates that express similar meaning for a given word, phrase or sentence.

1.2 Applications of Paraphrasing

In everyday life we use paraphrasing for many purposes: when rewording statements to explain ourselves better by simplifying or summarising them, when using our own words to write about a piece of news that we have recently read or when adapting to a different register, like a doctor does when he or she explains a disease to a patient. Paraphrasing can also be applied to many fields of NLP and has become very popular in the last years as a research line:

- In text summarisation, where simpler or shorter paraphrases can be generated from the source text, to preserve the original meaning while text is abbreviated. [Zha+18].
- In machine translation, to generate alternative translation candidates or translation references to increase variability [ZSW19] or to evaluate systems more accurately [ZLH06].
- In information retrieval, when expanding queries to improve the relevance of some documents that don't contain words of the query but contain related information in other words, also increasing coverage [Aga+18].
- In information extraction, when patterns are retrieved from documents they can be paraphrased in order to identify more patterns that have the same or similar semantic information.
- In question and answering, alternative paraphrased questions [SR19a] from the user can help to identify which question of the database match with the user's. Furthermore, it can be used to generate paraphrase answers to provide different replies each time.
- In any scientific or technical field where language has to be simplified, as in [Has16], where complex clinical jargon is simplified using paraphrasing to facilitate the comprehension of the diagnostics to the patients.
- In addition, there could be more applications that appear not to be investigated yet to the best of our knowledge. For example, in text-to-speech generation where voice is generated from written text, results can be paraphrased in order to adapt to spoken language. Also, voice assistant systems can use paraphrasing to generate alternate answers each time and sound more human.

As we can see, some of the applications have started to be investigated but there is still a lot of work left. Moreover, there are a lot of applications that have not been explored. To sum it up, paraphrasing appears to be one of the key factors to improve the performance of many current and future NLP systems.

1.3 State of the Art

Before the rise of deep learning in the fields of NLP, paraphrasing approaches were usually based on hand-crafted features mainly using databases containing paraphrase tables built with different paraphrase extraction techniques. Those involved knowledge-based methods (paraphrases extracted using human-constructed databases of semantic and lexical information about words and phrases) as well as lexico-syntactic methods (extraction based on words appearing in the surrounding context, in terms of N-grams frequency or TF-IDF counts, or on parsing the syntactic structure of the sentences and extracting words playing similar roles). Also filtering paraphrase candidates with some heuristics and performing hybrid methods that combine all lexical, syntactical and knowledge-based features in different ways [GVC13].

Taking into account the complexity of paraphrasing previously discussed, classic approaches seem to be very limited. Recently, deep learning has proven promising in making computers to deal with problems that are easy for humans to understand but difficult to formalise. That is the reason why, nowadays, some of state-of-the-art approaches are starting to explore deep learning techniques for paraphrasing, mainly concerning paraphrase generation and sentence representations. These techniques are mostly the same as the ones used in other natural language processing fields such as the use of recurrent neural networks, learning of fixed size representations of words and sentences into a vector space, and sequence-to-sequence models.

Sentence embeddings is one of the most popular techniques used to learn representations of features that can then help to deal with other tasks. In paraphrasing, some previous works have used sequence-to-sequence models, such as [Kir+15] and [Zha+18], to learn distributed representations of sentences that encode their semantic information. These representations are then tested in semantic relatedness and paraphrase identification tasks.

Another approach that has benefited a lot from sequence-to-sequence models is paraphrase generation. In the last years, [Pra+16] have explored more complex architectures and [Xu+18] started on models that focuses on the diversity of generated paraphrases. More recently, we have seen many improvements on these field. Ones exploring a more controllable generation of paraphrases, like [Che+19] and [Li+19], and others improving the decoding of neural models, such as [LMJ16] and [Kaj19]. Also another related fields, like question answering are using diverse paraphrasing in order to improve the performance of their models (see [JZS17] and [Dai+17]). Moreover, paraphrasing is becoming very important on specific domains like clinical reports, [Has16] and [SR19b] showed various applications.

In this regard, paraphrasing seems to start benefiting from deep learning in the same way as machine translation, question answering and other fields started to do some years ago. As we will see, one specific problem of paraphrasing is the lack of standard agreed metrics or reference corpus for testing, comparing or reproducing past tests.

1.4 Objectives

The main objectives of this project are:

- To explore a way of creating and evaluating sentence embeddings, that is to say, to learn semantic features in a generic task and then test them on paraphrasing related tasks.
- To investigate how these embeddings can be used to create a paraphrasing generation system based on neural network models exploring, as a start, sequence-to-sequence models.
- To explore and discuss the different ways of evaluating paraphrase generation models.

CHAPTER 2
Methodology

2.1 Summary

This chapter is structured mainly in two parts. Firstly, we will explain the background of our approaches: the sequence-to-sequence model in which our sentence embeddings and paraphrase generation models are based and the decoding of the sequences generated by this approach, needed for final paraphrase generation. Secondly, we will explain the differences that we introduce on sections 2.4 and 2.5. For sentence embeddings, we will use Skip-Though ([Kir+15]), that is a sequence-to-sequence model that uses one encoder and two decoders to predict the surrounding context of a given sentence. Having this as an input, for paraphrase generation we will introduce a new way of using a standard sequence-to-sequence model: instead of creating an encoder with random weights, we will use the encoder of Skip-Thought in order to take advantage of the previously learned weights.

2.2 Sequence to sequence

The main approach used in this project is the use of sequence-to-sequence models that we will describe in this section. These models are trained in order to learn how to convert one sequence from a given domain to a sequence of another domain. Even though these models are mainly used in the field of neural machine translation[SVL14], as paraphrasing can be considered a similar task, we find that they could also work for this task and others that we will describe later.

The specific sequence-to-sequence model used in our work is the encoder-decoder model based on recurrent neural networks, and more specifically the one made of Gated Recurrent Unit[Chu+14] (GRU). Assuming we have a sentence pair s_e and s_d (the source and the target sentence), the model is divided in three parts: the encoder, the decoder and the objective loss function.

The **encoder** is the first part of the model. It receives the $w_e^1, w_e^2, ..., w_e^N$ sequence of words of the source sentence s_e where N is the number of words in that sentence. At each time step t, produces a hidden state $\mathbf{h}_e^t$ that is the representation of the sequence formed by previous words and the current word $(w_e^1, ..., w_e^t)$. So the last state $\mathbf{h}_e^N$ is the representation of the full sequence. In order to obtain this representation, the encoder

model iterates over these equations:

$$\mathbf{r}_e^t = \sigma(\mathbf{W}_e^r \mathbf{x}_e^t + \mathbf{U}_e^r \mathbf{h}_e^{t-1} + \mathbf{b}_e^r) \tag{2.1}$$
$$\mathbf{z}_e^t = \sigma(\mathbf{W}_e^z \mathbf{x}_e^t + \mathbf{U}_e^z \mathbf{h}_e^{t-1} + \mathbf{b}_e^z) \tag{2.2}$$
$$\bar{\mathbf{h}}_e^t = \tanh(\mathbf{W}_e^h \mathbf{x}_e^t + \mathbf{U}_e^h (\mathbf{r}_e^t \odot \mathbf{h}_e^{t-1}) + \mathbf{b}_e^h) \tag{2.3}$$
$$\mathbf{h}_e^t = (1 - \mathbf{z}_e^t) \odot \mathbf{h}_e^{t-1} + \mathbf{z}_e^t \odot \bar{\mathbf{h}}_e^t \tag{2.4}$$

where σ is the hard sigmoid[1] function, $\odot$ is component-wise Hardamad product, $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are the weights matrices and the bias vector, $\mathbf{r}_e^t$ is the reset gate at time step t, $\mathbf{z}_e^t$ is the update gate, $\bar{\mathbf{h}}_e^t$ is the proposed state update and $\mathbf{h}_e^t$ is the hidden state.

The **decoder** is the second part of the model. It is a neural language model conditioned on the encoder last hidden state $\mathbf{h}_e^N$ and the previous words $w_d^1 ... w_d^t$ of the target sentence. It iterates over time steps following the same computations than the encoder, with the difference that its initial state $\mathbf{h}_d^1$ corresponds to the last state of the encoder $\mathbf{h}_e^N$:

$$\mathbf{r}_d^t = \sigma(\mathbf{W}_d^r \mathbf{x}_d^t + \mathbf{U}_d^r \mathbf{h}_d^{t-1} + \mathbf{b}_d^r) \tag{2.5}$$
$$\mathbf{z}_d^t = \sigma(\mathbf{W}_d^z \mathbf{x}_d^t + \mathbf{U}_d^z \mathbf{h}_d^{t-1} + \mathbf{b}_d^z) \tag{2.6}$$
$$\bar{\mathbf{h}}_d^t = \tanh(\mathbf{W}_i^h \mathbf{x}_d^t + \mathbf{U}_d^h (\mathbf{r}_d^t \odot \mathbf{h}_d^{t-1}) + \mathbf{b}_d^h) \tag{2.7}$$
$$\mathbf{h}_d^t = (1 - \mathbf{z}_d^t) \odot \mathbf{h}_d^{t-1} + \mathbf{z}_d^t \odot \bar{\mathbf{h}}_d^t \tag{2.8}$$

Given the hidden state $\mathbf{h}_d^t$, the probability of word w_d^t given the previous $t-1$ words and the encoder last state $\mathbf{h}_e^N$ is

$$P(w_d^t | w_d^{<t}, \mathbf{h}_e^N) \propto \exp(\mathbf{v}_{w_d^t} \mathbf{h}_d^t) \tag{2.9}$$

The **objective** loss function is the third part of the model. It aims at optimising the cross entropy (as minimising is equivalent to maximise log probability or maximise perplexity) of the target sentence conditioned on the last state of the encoder:

$$-\sum_t \log P(\hat{w}_d^t) P(w_d^t | w_d^{<t}, \mathbf{h}_e) \tag{2.10}$$

where $\hat{w}_d^t$ is the ground truth word of the target sentence at time step t.

Therefore, our encoder-decoder model produces a vocabulary distribution for each time step of the target sequence, that is conditioned of the last state of the encoder, and consequently, conditioned of the sequence of words of the source sentence.

2.3 Decoding

The decoding involves the process of converting the sequence of vocabulary distributions into text. In our model, the most probable word for each time step is conditioned on the previous predicted words. The decoder is a recurrent neural network and for each iteration it receives the previous output. This input is the embedding of the last predicted word, as we can see in 2.1. Also, the choice of the hypothesis word $\hat{w}^t$ for each time step t is conditioned on the previous chosen words as we show in the following equation (note that the condition on the last state $\mathbf{h}$ of the encoder is omitted for simplicity):

$$\hat{w}^t = \underset{v \in \mathbf{V}}{\operatorname{argmax}} P(w_v^t | \hat{w}^{<t}) \qquad\qquad 1 < t < N$$

[1] In Keras, sigmoid activation function for the update and reset gate is implemented with hard sigmoid, which is a non-smooth version of sigmoid usually used in neural networks for speed-up computations when precision is less important than speed.

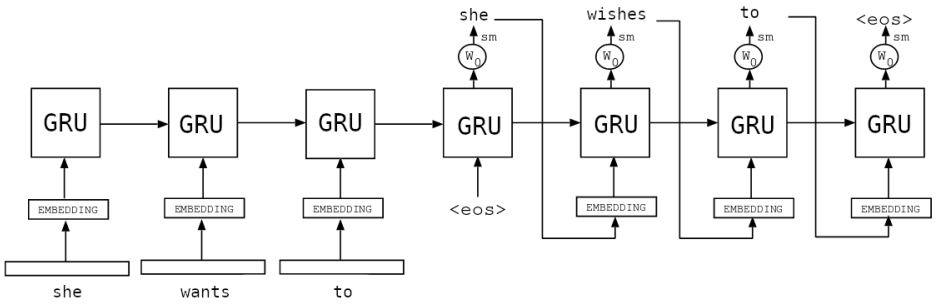

Figure 2.1: Illustration of the described sequence-to-sequence model performing a paraphrase generation task.

Where **V** is the probability distribution over the vocabulary.

Unlike other neural networks approaches, where the output that is considered correct is the one with the highest probability, the search is not as simple as picking the most probable word for each time step j conditioned on the most probable previous words. Due to the fact that the sequence that produces this greedy approach is not the optimal, we need to look for a more exhaustive search.

One of the most common search algorithms used for text generation in neural models is beam search. This algorithm keeps the k best paths during the search, exploring more candidates than the greedy approach. The probability distribution over words is computed conditioned on all the different paths that are being explored. So, for each time step and for each word, we have k different predictions based on the previous selected words. Next, we select the top-k words from all those possible candidates. As a result, we will have explored other paths that may look like less promising at the beginning but turn into the best path at the end.

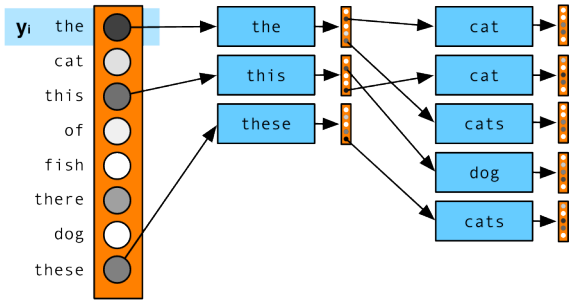

Figure 2.2: Illustration from [Koe17] of the different beams explored during the search.

2.4 Sentence Embeddings

There are many deep learning tasks that perform representation learning as a first step, before addressing directly their problem in order to learn representations that encode the nature of the data. Another common approach is to train a model that performs a generic task and then fine-tune it to target a more specific problem.

In NLP the most popular representation learning task is to learn word embeddings. It is performed by encoding words into a vector space (see [Ben+06]) so that words that have similar semantic meanings are placed in a nearby area . Usually, to this aim, neural networks trained to build the surrounding context of a given word (see [LM14]). Similarly, sentence vector representations can be learnt. This is very relevant to paraphrasing,

because if we have sentences of similar meanings encoded nearby, we can train models that use this vectors to identify them as paraphrases.

To clarify what similar vector representations means we will give some examples. As we can see in figure 2.3 word vector representations can encode different meanings, like semantic meaning of gender or syntactic information of verbs. A typical example is: if *man* vector is subtracted to *king* and then sum with *woman* we will obtain a vector whose values are very close to *queen*. In our case, we would like that the vector of the sentence *My father is rich* is close to, for example, *My father has a lot of money*.

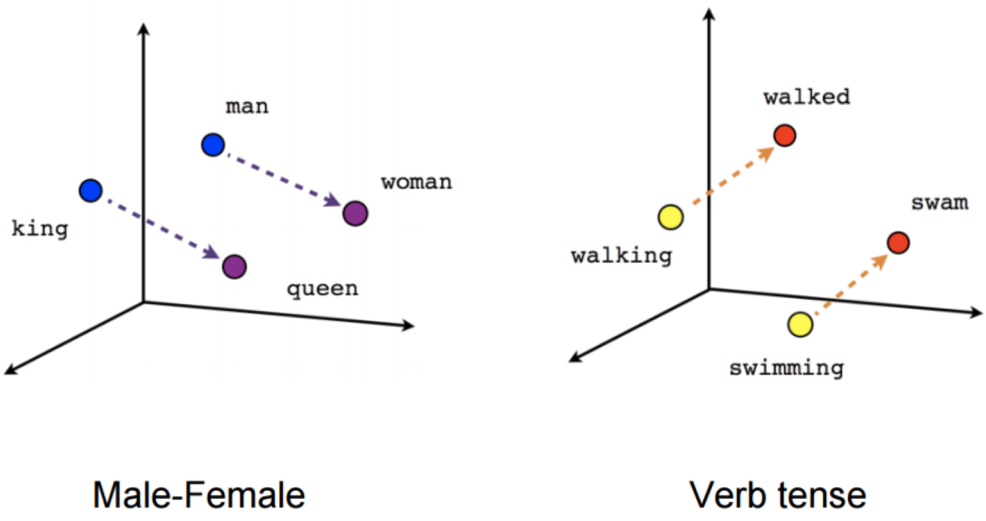

Figure 2.3: Illustration of word vector mappings.

In this project, we will explore one of the state-of-the-art approaches for learning sentence mappings to a vector space, that has proven reliable in some NLP tests, and more specifically paraphrase detection and semantic relatedness. But, to provide a different point of view, we will treat these sentence representations as a generic model that will be later fine-tuned to target a specific task: paraphrase generation.

2.4.1. Mean vectors

Our baseline approach to sentence embeddings will be some naive operations with word embeddings. We will see that, if we have pre-trained word vectors, a simple operation like the mean of all the vectors that are part of a sentence, can perform very well some paraphrasing tasks. It makes sense that, if we have encoded the semantic information of words, combining them would keep this information. The resulting model will be an encoder that has the word embedding layer as input and the output layer will be the mean of all word vectors from the input sequence. Thus we have a sentence representation result of the mean of word vectors that will be used as a baseline.

2.4.2. Skip-Thought

For sentence embeddings, we will deeply explore skip-thought vectors [Kir+15]. Inspired in skip-gram approach for words [Mik+13], skip-thought vectors encode sentences with a model that reconstructs its surrounding sentences. Unlike the word-level approach, each element of the sentence tuple (source, backward and forward, s_{i-1}, s_i, s_{i+1}) is made of a sequence of words instead of single one. Consequently the model becomes a sequence-to-

sequence model. Accordingly, our vector representation will be the last state of the first part of the model (the encoder), because this state would encode all of the information obtained through the iteration over all the words of the sentence.

In the original article, we can see some variations of this model and their results. We will explore a few more to evaluate the impact of learning of sentence embeddings applied to paraphrase generation. One of the explored variations will be the extension of the window context, since in the article only one backward and one forward sentences are considered. By widening the context, we expect that the sequence-to-sequence model will have more loss to optimise and, therefore, learn much more information.

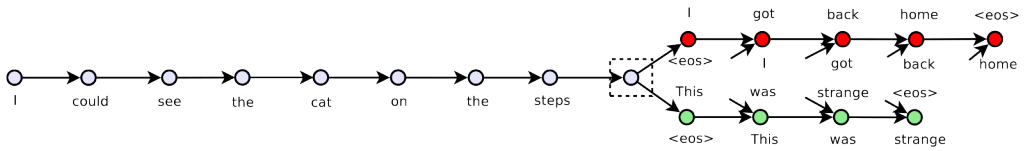

Figure 2.4: The skip-thought model with a window context of 1. In this example, we have the source sentence that is fed into the encoder *I could see the cat on the steps*, the forward sentence *I got back home* that the forward decoder tries to reconstruct, and the backward sentence *This was strange* that is reconstructed by the backward decoder. The dotted square marks what is considered to be the vector mapping of the source sentence.

Assuming we have a sentence tuple (s_{i-1}, s_i, s_{i+1} in the case of window context size 1 and $s_{i-2}, s_{i-1}, s_i, s_{i+1}, s_{i+2}$ of window of size 2), the model is described in three parts: the encoder, the decoders (one for each target sentence) and the objective function. Let the w_i^t describe the $t-th$ word for sentence i and $\mathbf{x}_i^t$ its word vector, the model is described as:

The **encoder**, as the first part of the model, receives the $w_i^1, ..., w_i^N$ sequence of words of the sentence s_i where N is the number of words in that sentence. At each time step t, produces a hidden state $\mathbf{h}_i^{t,e}$ in the same way as the encoder part described on section 2.2. So the last state $\mathbf{h}_i^{N,e}$ is the representation of the full sequence.

The **decoder**, as the second part of the model, is a neural language model conditioned on the encoder last hidden state $\mathbf{h}_i^N$. It consists of as many decoders as target sentences the model has. In the case of window context size 1, there is one decoder for the forward sentence s_{i+1} and another for the backward sentence s_{i-1}. And all of these decoders are conditioned on the last state of the encoder, and as a result, produce a vocabulary distribution at each time step for each target sequence of words.

The **objective** loss function to optimise is the sum of cross entropy of each context sentence conditioned on the last state of the encoder, so in the case of window context size 1 the sum will be:

$$-\sum_t \log P(\hat{w}_{i-1}^t) P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i) + -\sum_t \log P(\hat{w}_{i+1}^t) P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) \tag{2.11}$$

As the use of softmax function as output is very expensive to compute and the number of parameters grows dramatically when using large vocabularies, we will be constrained to train with with a small vocabulary of approximately 20,000 words. Then we can use the same approach described in the original article to perform a vocabulary expansion. This vocabulary expansion is a linear mapping between a set of pre-trained word representations and the vocabulary of the model. To learn this linear mapping, we will train a linear regression model using the vocabulary that intersects with the pre-trained embeddings. Then, the vocabulary from the pre-trained word embeddings that is

not on the skip-though vocabulary will be mapped to that space. This will let our model have a lot more words in the vocabulary.

At the end of the training, the decoders of this model are no longer required, since we are only interested on sentence representations, as showed in figure 2.4. As a result, we will obtain the encoder part, that receives a sentence and produce a multi-dimensional vector that we will consider the semantic representation of that sentence. The main reason why this approach has been chosen, is because it can be used to train a sequence-to-sequence model for paraphrase generation that is initialised with the weights of this pre-trained encoder, as we will see in the next section. Also because training is unsupervised making the compilation of data much easier and avoiding the limitation of other models like InferSent[Con+17] that need labelled data to be trained.

2.5 Paraphrase Generation

Paraphrase generation is a task that, given a sentence, phrase or word, aims at producing one or more sentences, phrases or words that have roughly the same meaning. This can be viewed as a similar task to machine translation, but where the model is "translating" to the same language. Paraphrasing is some kind of "monolingual translation" so, we assume that it can be performed by a model inspired in machine translation (see 2.1). Sequence-to-sequence models are the state-of-the art techniques in neural machine translation and Skip-Thought is a similar approach. One of the goals of our work is to take the advantage of the similarity of the architecture of Skip-Thought with neural machine translation models. In order to create a sequence-to-sequence model whose encoder weights are pre-trained, we take advantage of modern learning libraries like Keras ([Cho+15]) to train and re-use, more easily, the encoder of Skip-Thought. Therefore, the decoder of the paraphrase generation will act as a target language model conditioned on the vector representation of the source sentence. Then, this sentence representation will be pre-trained on a generic task which has proven to perform well on paraphrasing related tests.

The optimised loss function will be the same as described on the previous section 2.2. Then, the Truncated Back-propagation Through Time [Bro17], will use the Adam optimiser to optimise this loss function. That will result on a training algorithm that does not need its parameters to be adjusted, since its parameters will adapt while training evolves. Finally, the decoding process will be done in the same way as described in 2.3, using beam search.

For our purposes, the paraphrase generation task will work at phrase level, that can be considered a simplification of sentence level. Since our sentence encoder model works with word sequences, it is also capable of encoding phrases. Basically, phrases can be treated exactly as short sentences, that is, as sequences of words. The decision to work at phrase level will allow also re-using the biggest and most widely used corpus available in the literature (PPDB [GVC13]).

CHAPTER 3
Experiments

3.1 Experimental framework

First of all, we will explain the overall structure of the experiments, and then, further details will be given in the next sections. In general, the followed steps have been:

1. Build our implementation of Skip-Thought.
2. Train various Skip-Thought models varying different hyper-parameters, like the number of neurons, the size of the window or the training corpus.
3. Compare all the models with the baseline and the literature, in order to see if we can improve the original Skip-Thought.
4. Use the pre-trained Skip-Thought encoder to create a paraphrase generation system.
    - Compare it to a baseline with random weights, in order to see if pre-trained weights from Skip-Thought can improve the baseline performance.
    - Compare it with other paraphrase generation systems from the literature.

3.2 Corpora

For the training and evaluation of our models, we used different corpora that are described below. A summary of the corpora, what were they used for and their sizes, is shown in table 3.4.

OpenSubtitles [Pie16] is a large multilingual data set made of subtitles of movies, TV series and documentaries from the OpenSubtitles.org website. It was chosen because it has a high variety of vocabulary and short sentences. Since we are interested in phrases, short sentences were deemed to be interesting. Also because sentences in this corpus preserve the order of the subtitles and we considered it very useful to extract the context of each sentence to train our Skip-Thought models. We extracted 20 million sentences in English for training.

BookCorpus[Zhu+15] is a data set of books in English crawled from the web. The original corpus is no longer distributed, so we used the crawler published in the repository to collect our own corpus with the provided URLs in the repository. This data set was used in the original paper of Skip-Thoughts and it also has continuous text that can be used to extract the context for each training sample. The size of the corpora crawled using the URLs from the toolkit repository is 8.6M sentences.

Microsoft Research Paraphrase Corpus[DQB04] is a collection of pairs of sentences tagged as paraphrases (positive = 1) or not (negative = 0). It consists of 4,076 sentence pairs for training and 1,725 pairs for testing and 66% of the sentences are positive.

MSRPC samples
The results will be published the July 10 issue of the journal Nature. The results appear in Thursdays issue of the journal Nature. 1
Physicians who violate the ban would be subject to fines and up to two years in prison. Physicians who perform the procedure would face up to two years in prison, under the bill. 1
The broader Standard & Poor's 500 Index rose 3.42 points, or 0.34 percent, to 1,007.84. The technology-laced NASDAQ Composite Index .IXIC was down 1.55 points, or 0.09 percent, at 1,744.91. 0
Qanbar said the council members would possibly elect a chairman later Sunday. US authorities have said the council would include 20 to 25 members. 0

Table 3.1: Pairs of sentences and its ground truth (1 means true paraphrase and 0 means not a paraphrase).

SICK[Mar+14](Sentences Involving Compositional Knowledge) data set comes from a semantic relatedness task from SemEval 2014. It consists of 4,500 sentence pairs for training, 500 for development and 4,927 for testing. These pairs are annotated with a 5-point rating scale that indicates the degree of semantic relatedness between each sentence.

SICK dataset samples
Five children are standing in front of a wooden hut Five children are standing in a wooden hut 4.2
The equipment in front of the blond dancing girl is sound A girl in white is dancing 2.7
A man is playing an electric guitar A group of people is holding drinks and pointing at the camera 1
Four children are doing backbends in the gym Four girls are doing backbends and playing outdoors 3.8

Table 3.2: Pairs of sentences with their score; higher scores mean more semantic relationship.

Paraphrase Database[GVC13] is a multilingual data set, that was automatically collected, consisting of syntactical, lexical and phrasal paraphrase pairs. We sampled 5 million paraphrases from the XXXL databases of lexical and phrasal paraphrases. For sampling, we used the PPDB2.0[PRJ15] score provided for each paraphrase pair, in order to filter the ones that have the highest probability of being paraphrases. The score threshold was picked manually (3.9 for phrasal and 3.9 for lexical). This threshold allowed us to get enough data for training and samples that are not very rare or erroneous paraphrases.

PPDB samples	
come to an end	are completed
action on the	measures in the
preferred	demanded
rotates	flips
an in-depth discussion	detailed discussion
commissioned	perpetrated
allows us	also helps
and enables	, thus giving
been authorised	therefore, the
don't you think	especially

Table 3.3: Some lexical and phrasal samples from the Paraphrase Database. The top section shows examples from the database filtered by score higher than 3.9, and the bottom section samples from the rest of the corpora. This example illustrates how noisy can be this corpus if it's not filtered.

3.3 Training

3.3.1. Preprocessing

For all tasks, minimal preprocessing was done. We created different data generator classes to abstract the logic of reading the corpora, preprocessing them and providing batches to the training models, since our models tend to use all GPU memory and all the data cannot be loaded directly. The tasks performed by these classes behave as follows:

- Read the entire training corpus and load it on RAM.
- Tokenise the data using Keras[Cho+15] and store it in an index.
  - It builds a dictionary of the corpus sorting the words by frequency.
  - It transforms text into sequences of indexes from the vocabulary, tagging the words below the vocabulary threshold as "unkown". Please note that filters can be removed in order to have a dummy tokeniser that splits words by spaces, if the data comes tokenised from an external tokeniser.
- Yield the batches needed during training. Keras training method requests an item in every step of the training and the class gathers the data from its index and creates a batch of tuples. Skip-Thought receives as input a batch of tuples (triplets in the case of window 1) of source sentence and target sentences (forward and backward), and outputs the target sentences (backward and forward). The paraphrase generation model is very similar to Skip-Thought but taking only one sentence as input and output.

The only sophisticated method of preprocessing that has been explored is byte pair encoding (BPE) [SHB15]. BPE encodes rare or unseen words by means of sequences of subword segments. These subword segments are learned on the training corpus and the most frequent ones are added to the vocabulary. As a result, the model is trained without unknown words ("theoretically complete coverage"), so the language model is forced to explain all unkonw or rare words in terms of subword segments. The use of BPE has proven to be very useful in neural machine translation and it is is nowadays considered a standard preprocessing method. Moreover, it can be easily applied or ignored using just one line of bash scripting. Despite this preprocessing was tried, no results were reported since we don't encounter any relevant results.

3.3.2. Building Skip-Thought models

First of all, the Skip-Thought toolkit was downloaded in order to reproduce the experiments showed in the paper. The pre-trained models are available to download and test them, but the corpus used to train the models is no longer available and has to be crawled with a toolkit. This made the experiments not fully reproducible.

After verifying that the toolkit works with another corpus (OpenSubtitles) and the tests produce some results that prove that the model is working, we decided to re-implement the code in Python 3 and Keras[Cho+15] as the available toolkit was using Python 2 and Theano, making it very hard to use. On the one hand, Python 2 support will end by 2020, just a few months after launching our experiments. On the other hand, Theano is a low-level library (making coding and maintenance more complex) and it has lower performance than others like Tensorflow, the default Keras backend. Thus, implementing the toolkit in Keras and Python 3 would allow for a much easier maintenance and extension of the code and functionalities.

More in detail, implementing the toolkit in Keras brought the following benefits:

- Update the code to a more recent library and version of Python.
- Improve the extensibility, readability and maintainability of the code, since Keras model architectures are much more explicit with many fewer lines of code.
- Ability to re-use the trained models along with others or plug-in different types of decoders or encoders easily (re-use for paraphrase generation).
- Improve the overall performance and reduce training time.
- Ability to create multi-GPU models with one more line of code.

The code made for the project is available at [Zar19].

In Keras, we use categorical cross entropy as the loss function to optimise, as minimising the cross entropy is the same as maximising the log-likelihood. More specifically, we use sparse categorical cross entropy, which can compute cross entropy with a given vector of indexes that represent the position of the word in the vocabulary. This sparse computation was the key point to train almost 4 times faster than before by explicitly avoiding the use of one-hot vectors, that take a lot more space, and the softmax function, much more expensive to compute.

The training of the Skip-Thought models was executed in a Nvidia GTX1080 GPU and the gradient descent used was Adam with the default parameters of Keras (learning rate of 0.01 with no decay), batches of 128 for context window size 1 and 64 for size 2, since the model with context window size 2 is bigger with the risk of processing resulting in an out of memory error. The tested hyper-parameters were:

- the number or neurons of the recurrent layer of the encoder and the decoder.
- the type of recurrent layer used (LSTM or GRU).
- the size of the context window (the number of forward and backward sentences to predict)
- the different training corpora.
- the sharing or not the vocabulary matrix of the decoders.

As we can see in the figure 3.1 the training is far from convergence and it is overfitting, these could be due to the simplicity of the encoder and the decoder, the only have one layer. We can also observe that the model with window 2 evolves in the same way as the one with window 1, but it has more distance between the train loss and the development loss because it has 4 decoders as output (and therefore, 4 loss sources). Another remarkable aspect is that the model stops improving on almost on epoch 27, meaning that only approximately 3.8M sentences were seen (much less than the total corpus), as batch size is 128 sentences and steps per epoch are 1,000.

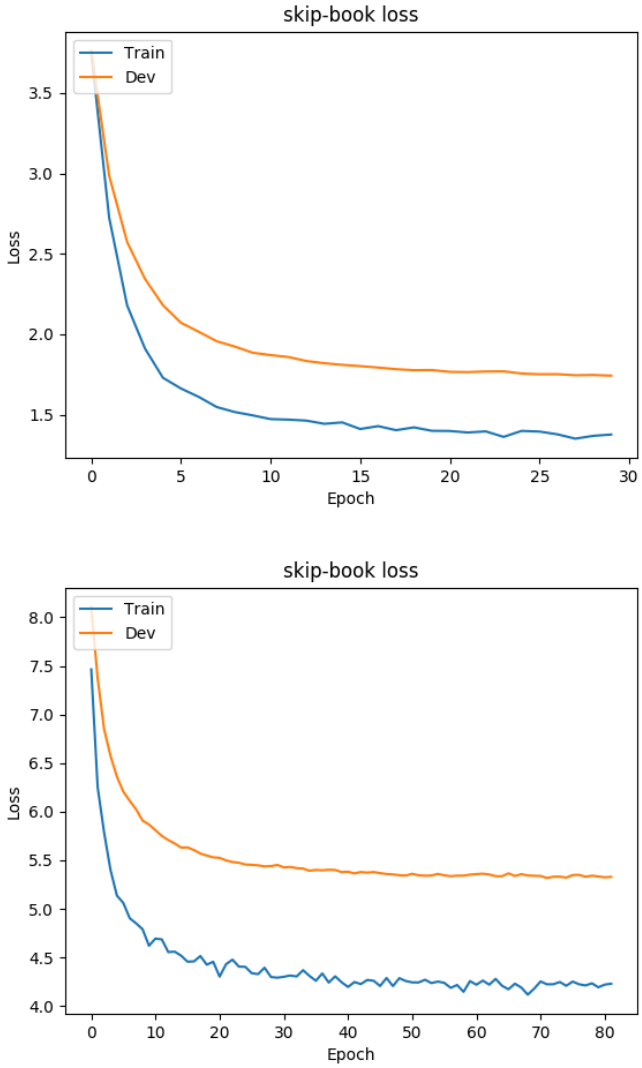

Figure 3.1: The evolution of cross-entropy loss during training for training set and development set. At the top we can see the model of a context window of 1 and at the bottom a context window of 2.

3.3.3. Encoder

An encoder class was created to handle the encoding of sentences to vector representations using the last state of the Skip-Thought encoder. This class performs the following tasks:

1. Load the pre-trained Skip-Thought model and the pre-trained embedding file from GloVe[PSM14].
2. Find the vocabulary that is shared between the word embeddings file and the Skip-Thought vocabulary.
3. Train a linear regression using the shared vocabulary as training samples, with the Scikit-learn toolkit.
4. Apply linear regression to all the vocabulary from the pre-trained embeddings that is not in the Skip-Thought vocabulary.
5. Create a new model using the configuration and the weights from the Skip-Though model, but only the encoder part.
6. Append all the new vectors to the embedding layer of the new model.
7. Use the last state of the GRU as the output of the new model.

At the end, the encoder class provides a model that receives a sentence, tokenizes it and encodes it. This class is then used on the embedding tests and also to create the paraphrase generation model.

This encoder class also is able to create an encoder which performs the mean vectors approach. It loads the embedding file and creates a Keras model that has the embedding layer with these vectors as weights. All word embeddings from a given sentence are fed into a lambda layer that computes the mean. Thus, it provides the same output shape of the other encoder model, that is the mean of word embeddings instead of the last state of a recurrent layer.

3.3.4. Paraphrase Generation

For the generation phase, we used the encoder explained in section 3.3.3 to set our base weights and we appended a new decoder. The new decoder is made of a recurrent layer of the same type and size than the encoder, and a fully-connected layer as vocabulary distribution at its output. Since the decoder weights are not pre-trained, meaning that weights of both parts of the model are unbalanced, we performed a two phase training (well-known technique in transfer learning for neural networks). On the first phase, in order to propagate gradient only to the decoder weights, the encoder weights (both recurrent weights and embedding weights) are frozen. This phase takes only about oen epoch because at the beginning the gradient is much bigger and the parameters are learned much faster. On the second phase, the encoder weights are unfrozen and all the model weights are trained jointly, allowing the encoder to be fine-tuned to the specific task.

3.4 Evaluation

3.4.1. Embedding tests

We present two experiments to test the quality of the trained word vectors. The first experiment has to do with the semantic relatedness task from the SICK dataset [Mar+14]. The aim of this task is to create a regressor that tries to predict a score for each pair of sentences, a score that represents how similar are they in terms of semantic relationship. The second experiment has to do with the paraphrase detection task from the MSRPC

[Com]. The aim of this task is to produce a classifier that identifies if a pair of sentences are paraphrase or not.

In order to demonstrate the quality of the trained word vectors, we tested them in the same way as showed in the literature. To train the models to perform these tests, only two features were used, computed directly from the features learned on the sentence embeddings. In the same way as [SSM15] did, given two sentences we compute their u and v vectors and then their absolute difference $|u - v|$ and their component-wise product $u\dot{v}$. With these features, linear regression is trained for predicting the score as in the SICK task and a classifier is trained to be able to classify paraphrases as in the MSRPC task.

The reason why the Skip-though approach is very powerful, is because it let us to perform two experiments with features that are learned on a completely different task. First of all, a task that does not specify the information of paraphrase identification or semantic relatedness. And then, perform at par of other approaches that are built explicitly for those tasks or perform a supervised feature learning, instead of Skip-Thought that is unsupervised.

To perform these tests we used the same scripts that are provided in the Skip-Thought toolkit, with some modifications to update the code and read the data more easily. Also, in order to be able to apply BPE if the trained model required it, we have re-formatted the two data sets used, removing the columns that are not relevant for these experiments.

The metrics used in these tests are:

- For the semantic relatedness task:
  - Pearson's r correlation coefficient, that measures the covariance of two variables in terms of quantity, but independently from the scale of the variables. A value of 0 means no correlation and +1 or -1 mean that are completely related negatively or positively.
  - Spearman's ρ correlation coefficient, that measures if the values of two variables are replaced in the same order. 0 means no correlation and +1 or -1 mean positive or negative correlation.
  - Mean squared error (MSE) is a non-linear measure of the difference between two variables that penalises more when the difference is higher. It is a quite often used measure of error in regression problems. The two correlation measures are used in order to verify if an improvement of this error also means more correlation.
- For the paraphrase detection task:
  - Accuracy: the most used metric in classification tasks. It measures the percentage of well classified samples.
  - F-measeure, also called F1: a measure that is considered the harmonic mean between precision and recall.

The **baseline** for these tests will be the mean-vectors approach described on the section 2.4.1, because it is a naive model that does not require the training phase that Skip-Thought performs. Other naive approaches that perform different word vectors operations like addition, subtraction or component-wise product were explored, but all of them resulted on models that do not learn during the training process and they were, therefore, discarded.

3.4.2. Paraphrase generation tests

The models we compared to in the paraphrase generation task are:

- Diverse Paraphrasing Generation (D-PAGE)[Xu+18]: this model focuses on the diversity of generated paraphrases. To increase the diversity it uses several decoders in its sequence-to-sequence model, each of them has a different reordering pattern to produce different types of paraphrases. The number of encoders is appended at the end of the model.

- Residual Long Short-term Memory (LSTM)[Pra+16]: a sequence-to-sequence model where its encoder and decoder are made up of several stacked recurrent LSTM layers (ours only has one). Also these layers have residual connections between them, this means that the hidden state of a given layer is a combination between its last hidden state and the input of the previous layer.

The metrics used for the paraphrase generation tests are:

- BLEU (bilingual evaluation understudy)[Pap+01]: this score is considered as a *de facto* standard in some NLP tasks like machine translation. It is also the only metric that is common to all paraphrase generation articles reviewed. It counts the percentage of all possible n-grams (usually unigrams to 4-grams) from a reference sentence that are present on a hypothesis sentence. Despite being widely used, BLEU has been highly criticised due to its limitations [Tat15]. We see that the main limitation for paraphrasing is that BLEU does not consider the meaning of the words or alternatives so, for example, for a source phrase *responses to*, if the reference paraphrase is *replies to the* and the hypothesis is *replies to*, the BLEU score between the hypothesis and the reference will be 60 despite being a perfect paraphrase. And worse, if the hypothesis is *answers to* the score will be almost 0, despite being also a perfect paraphrase.

  So, we deem important complementing this metric with another one.

- Greedy matching between word embeddings[RL12]: this method computes a score based on the similarity of word embeddings of the hypothesis and the reference. Thus if we have a phrase that is very different to the reference in terms of n-grams, it can be scored quite well if the semantic relationship is high. For the example above mentioned: *replies to* in front of *replies to the* has a 7.7 score and *answers to* in front of *replies to the* has a score of 5.3, which is lower but it's much closer than BLEU scores. The pre-trained word vectors used for the test are Google News embeddings[1] trained with Word2Vec at [MLS13], that is the same used in [Pra+16].

Automatic evaluation is very useful to test in a cheap and quickly way and to have an impartial metric, but it is limited when evaluating deeply if the hypothesis are good or not or what the source of errors is. As we have seen above, not all good scores correspond to good hypothesis and vice versa. Also a humans can evaluate in many ways that automatic metrics cannot do (i.e. they can take into account the whole structure phrase or sentence, unlike BLEU). So, we decided to perform human evaluation also.

To perform this evaluation, we choose tree human evaluation metrics that are used in [Has16]. Two people evaluated a subset of 100 phrases randomly selected from the test set in terms of:

[1] The pre-trained vectors are available at https://github.com/mmihaltz/word2vec-GoogleNews-vectors

- Semantic relatedness: how much of the original meaning is retained on the candidate phrase.
- Novelty: how much the paraphrase is considerably different from the source.
- Grammaticality: how much the candidate paraphrase can be considered as correct according to English grammar.

Each score can take values from 1(very bad) to 5(very good), then the mean between the two human judgements is calculated and the final score is an average of the three metrics for each model (the model type used is hidden to evaluators).

The **baseline** of these tests will be a sequence-to-sequence model, basically the same that is created with Skip-Thought but with all the weights initialised as random. The comparison with this model will show us if the pre-trained encoder can improve the performance of the sequence-to-sequence model.

Corpus	Description	Train size	Dev size	Test size	Used for
OpenSubtitles	TV and movie subtitles	20M	10K	-	Train sentence embeddings
BookCorpus	Publicly available books	8.6M	10K	-	Train sentence embeddings
MSRPC	Praphrase identification	4K	-	1.7K	Test sentence embeddings
SICK	Pairs of sentences and their semantic relationship	4.5K	0.5K	4.9K	Test sentence embeddings
PPDB	Pairs of phrasal and lexical paraphrases	5.3M	10K	4K & 100	Train and test paraphrase generation

Table 3.4: Summary of all the corpora used.

CHAPTER 4
Results

4.1 Embedding tests results

We present the results for the SICK and MSRPC tests in this section. These are shown in tables 4.1 and 4.2 below. Each table is divided in two sections: in the upper section our results and in the lower section the original Skip-Thought and other state-of-the-art models results that are shown to illustrate in which range the scores are. In both tables, our methods read as follows: *uni-skip* indicates the main name of our method (unidirectional Skip-Thought), n indicates the number of neurons used on encoder and decoder (2400 in all cases), w indicates the size of the context window (1 or 2) and finally the name of the training corpus is shown (Books or Open-Subtitles).

In table 4.1, reporting SICK results from the literatures, we can find the following methods:

- Mean vectors: the mean of word vectors, that is, presumably, the same that our baseline.
- The three Skip-Thought[Kir+15] models (unidirectional, bidirectional and combined).
- Dependency-Tree LSTM[SSM15]: a model made for this types of tasks.
- InferSent[Con+17]: the best performance. It also learns generic sentence representations but using labelled data.

In table 4.2, reporting the MSRPC results, we can find the following methods:

- Cosine similarity of tf-idf weightening vectors[Com] (this is the original baseline of the task).
- A combination of lexical and semantical features.
- The three Skip-Though[Kir+15] models: unidirectional, bidirectional and combined.
- TF-KLD[Com]: matrix factorisation with supervised re-weightening, that has the best result on the task.

Observing these results, we can draw some conclusions:

- In the SICK tests, our approach outperforms the baseline results.
- In the MSRPC test, the baseline mean-vectors approach outperforms our results.

Method	$r \uparrow$	$\rho \uparrow$	**MSE**$\downarrow$
uni-skip n2400 w1 books	0.7275	0.6569	0.4974
uni-skip n2400 w1 opensub	0.5640	0.5400	0.6371
uni-skip n2400 w2 books	0.7690	0.6996	**0.4234**
uni-skip n2400 w2 opensub	0.7347	0.6820	0.4892
mean-vectors (baseline)	0.6408	0.6348	0.61
mean-vectors [SSM15]	0.7577	0.6738	0.4557
uni-skip [Kir+15]	0.8477	0.7780	0.2872
bi-skip [Kir+15]	0.8405	0.7696	0.2995
combine-skip [Kir+15]	0.8584	0.7916	0.2687
Dep-Tree LSTM [SSM15]	0.8676	0.8083	**0.2532**
InferSent [Con+17]	**0.883**	-	-

Table 4.1: SICK test results. The metrics are Pearson(r) and Spearman(ρ) correlation coefficients (higher means better) and mean squared error (lower means better). In the upper section there are our approaches, in the lower section the ones from the literature.

Method	Acc$\uparrow$	**F1**$\uparrow$
uni-skip n2400 w1 books	66.78	75.52
uni-skip n2400 w1 opensub	65.44	74.81
uni-skip n2400 w2 books	68.92	77.13
uni-skip n2400 w2 opensub	66.95	75.11
mean-vectors (baseline)	74.9	**82.40**
Cosine sim [Com]	65.4	75.3
Lex&Sem feat. [Com]	76.6	79.6
uni-skip [Kir+15]	73.0	81.9
bi-skip [Kir+15]	71.2	81.2
combine-skip [Kir+15]	75.8	83.0
TF-KLD [Com]	80.4	**85.9**

Table 4.2: MSRPC test results in terms of accuracy and f-measure. In the upper section there are our approaches, in the lower section the ones from the literature.

- In both tests, we could not reproduce the the state-of-the-art results, especially on the MSRPC test. This can be due to the fact that our models were trained with much less data and time than the original Skip-Thought model (they used 70 million sentences and more than a week).

- We can see an improvement in our models, in both tests, when the window of context size 2 is used. This confirms our initial hypothesis: the bigger the context, the more the information it can provide to the encoder, and the more it can learn.

4.2 Paraphrase generation tests results

Table 4.3, reporting paraphrase generation results, is also divided in two sections like the ones for the embedding tests results. It must be highlighted that the sampling of the PPDB data set is not the same between the experiments of the cited papers and ours (see explanation below), so the results are not completely comparable. Also BLEU scores are not completely comparable as they depend on some parameters (like tokenisation and

casing) and these are not specified in the original papers. For computing BLEU metric we used SacreBLEU[1] toolkit[Pos18] with default parameters.

In the table, we can find the following methods that correspond to the mentioned papers:

- Diverse Paraphrasing Generation (D-PAGE)[Xu+18]: for training and testing they randomly sampled 4.5M and 0.5M sentences respectively, from the XXXL database. They don't seem to be filtering by score, and that means that their training corpora could be very noisy (as we can see in table 3.3). Also, they only use the phrasal database and not the lexical one.
- Residual Long Short-term Memory (LSTM)[Pra+16]: for training and testing they used a corpus which is similar to us, sampling the lexical and phrasal PPDB L size. This resulted in 5.3M paraphrases from which 90% are selected for training and 20K randomly sampled from the remaining 10% are retained for testing.

Method	BLEU	**Emb Greedy**
seq2seq n2400 (baseline)	10.63	2.61 $\pm$ 0.00
seq2seq + skip n2400 w1 books	**14.82**	**3.06** $\pm$ 0.00
seq2seq + skip n2400 w1 opensub	11.06	2.91 $\pm$ 0.00
D-PAGE-2 [Xu+18]	16.5	-
D-PAGE-4 [Xu+18]	15.4	-
D-PAGE-8 [Xu+18]	14.1	-
Residual LSTM [Pra+16]	20.3	*34.77**

Table 4.3: Results on PPDB test set.

* The metric showed in this paper does not correspond to the metric that we have computed. We used the same script and the same embeddings that they mention in the paper, but results on the paper seem to be in a different scale. We computed the embedding metric assuming that the hypothesis are all perfect (the same as the reference) and the maximum score obtained was 9.6, so we don't know how they could get to the reported value. Also on the paper that originally proposed this metric, the values vary between 0.35 and 0.20. Thus, the values are not comparable.

As we can see in table 4.3 our proposed architecture of a sequence-to-sequence model with a the pre-trained Skip-Thought encoder outperforms the baseline in BLEU and embedding. We would like to remark that the baseline and the other two proposed models have all the same number of parameters (varying a bit on the vocabulary, but negligible). Despite this improvement, the proposed model is bit far from the state-of-the-art. In terms of BLEU is far from the Residual LSTM approach, but we have to consider that this model has a more complex architecture. For the D-PAGE model, BLEU values are similar but we need to take into account that this model is focused on the diversity of its hypothesis.

Finally, we report the results comparing the baseline and our best system according human evaluation tests. First, lets take a look to some random sampled paraphrases extracted from it in table 4.4. This table shows that, despite the model does not guess correctly the target paraphrases, most of the paraphrase preserve the original meaning and are grammatically correct. Furthermore, there are examples where the hypothesis meaning is closer to the source meaning than the target. For example: *that's just terrific*

[1]The aim of this toolkit is to provide clear reports of BLEU scores that can be reproducible.

Source	Hypothesis	Target
deputies from	representatives of	delegate to
be concluded	conclusion,	be celebrated
form an integral part	are an integral component	constitutes integral part
this is perfect	that's just terrific	it's fine
conditions were	terms and conditions governing	requirement as
president said	chairman stated	, the head of
enabled us	have allowed us	has made it possible
vacant posts	vacant positions	vacant positions
requires the	there is a need	requiring the
applies only to	is restricted to	applies to
of deciding	to take a decision	-

Table 4.4: A sample of paraphrases from the test and the hypothesis produced by the model.

is closer to *this is perfect* than *it's fine*. Those are the things we are trying to test more accurately when using the embeddings metric, not focusing on the exact n-grams of a hypothesis paraphrase but rather on the meaning.

Model	Meaning	Novelty	Grammaticality	Average
seq2seq	3.11	2.29	2.84	2.74
seq2seq + skip n2400 w1 books	3.70	2.87	4,31	**3.54**

Table 4.5: Results of the human evaluation tests.

According to the results on the human evaluation reported in table 4.4 we can see clearly that the proposed model outperforms the baseline. The *grammaticality* is quite close to 5 (the highest score) in our best system and the *meaning* is also significantly better than in the baseline. Only *novelty* is basically on par with the baseline. The reason behind it can be that we did not made any modification on the model to focus on diversity, like [Xu+18] did.

CHAPTER 5
Conclusions

This thesis described how Skip-Thought vectors, trained on a generic task that is not directly related to paraphrasing, can be used to initialise the encoder part of a paraphrase generation model, and improve its performance. These vectors were also tested on paraphrasing related tasks, and despite not being able to perform at the state-of-the-art level due to the use of fewer data, we expect that we are on track of achieving this level in future. By the time, we only tested a few variations of this model could bring. One of the possible models to test in the future, could be [Xu+18] but using the initialisation we introduced, training our proposed model using the LSTM architecture proposed by [Pra+16], or exploring other ways of decoding. We also created a toolkit that is publicly available and can be used, more easily than the original (because of newest code and libraries), to extend its functionality and/or to test different architectures.

We showed that paraphrase generation addressed with deep learning is growing faster, and it needs more consistent ways of testing it. Because of the lack of standard metrics, corpora and the reproducibility of the experiments, this field can't benefit of more reliable comparisons. Compared to neural machine translation, these are experiments are harder to accomplish.

Bibliography

[Pap+01] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *ACL*. 2001.

[DQB04] Bill Dolan, Chris Quirk, and Chris Brockett. "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources". In: *Proceedings of the 20th International Conference on Computational Linguistics* (Jan. 2004). DOI: 10.3115/1220355.1220406.

[Ben+06] Y. Bengio et al. "Neural Probabilistic Language Models". In: vol. 3. May 2006, pp. 137–186. DOI: 10.1007/3-540-33486-6_6.

[ZLH06] Liang Zhou, Chin-Yew Lin, and Eduard Hovy. "Re-evaluating Machine Translation Results with Paraphrase Support." In: Jan. 2006, pp. 77–84. DOI: 10.3115/1610075.1610087.

[Ho+12] ChukFong Ho et al. "Extracting lexical and phrasal paraphrases: a review of the literature". In: *Artificial Intelligence Review* (2012). DOI: 10.1007/s10462-012-9357-8.

[RL12] Vasile Rus and Mihai Lintean. "A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics". In: June 2012, pp. 157–162. DOI: 10.1007/978-3-642-30950-2_116.

[GVC13] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. "PPDB: The Paraphrase Database". In: *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*. 2013. URL: http://www.aclweb.org/anthology/N13-1092.

[MLS13] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. "Exploiting Similarities among Languages for Machine Translation". In: *ArXiv* abs/1309.4168 (2013).

[Mik+13] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[Chu+14] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[LM14] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[Mar+14] Marco Marelli et al. “SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment”. In: *SemEval@COLING*. 2014.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[Cho+15] François Chollet et al. *Keras*. https://keras.io. 2015.

[Kir+15] Ryan Kiros et al. “Skip-Thought Vectors”. In: *arXiv preprint arXiv:1506.06726* (2015). arXiv: 1506.06726.

[PRJ15] Ellie Pavlick, Pushpendre Rastogi, and Chris Callison-Burch Juri Ganitkevitchand Ben Van Durme. “PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Beijing, China: Association for Computational Linguistics, 2015.

[SHB15] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *CoRR* abs/1508.07909 (2015). arXiv: 1508.07909. URL: http://arxiv.org/abs/1508.07909.

[SSM15] Kai Sheng Tai, Richard Socher, and Christoper Manning. “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. In: (2015). DOI: 10.3115/v1/P15-1150.

[Tat15] Rachel Tatman. *Evaluating Text Output in NLP: BLEU at your own risk*. 2015. URL: https://towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213.

[Zhu+15] Yukun Zhu et al. “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

[Has16] Sadid Hasan. “Neural Clinical Paraphrase Generation with Attention”. In: (2016).

[LMJ16] Jiwei Li, Will Monroe, and Dan Jurafsky. “A Simple, Fast Diverse Decoding Algorithm for Neural Generation”. In: (Nov. 2016).

[Pie16] Jörg Tiedemann Pierre Lison. “Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles”. In: *In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)* (2016). arXiv: 1804.01768.

[Pra+16] Aaditya Prakash et al. “Neural Paraphrase Generation with Stacked Residual LSTM Networks”. In: *COLING*. 2016.

[Wie+16] John Wieting et al. “Towards Universal Paraphrastic Sentence Embeddings”. In: (2016).

[Bro17] Jason Brownlee. *How to Prepare Sequence Prediction for Truncated BPTT in Keras*. 2017. URL: https://machinelearningmastery.com/truncated-backpropagation-through-time-in-keras/.

[Con+17] Alexis Conneau et al. "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 670–680. URL: https://www.aclweb.org/anthology/D17-1070.

[Dai+17] Bo Dai et al. "Towards Diverse and Natural Image Descriptions via a Conditional GAN". In: (2017).

[JZS17] Unnat Jain, Ziyu Zhang, and Alexander G. Schwing. "Creativity: Generating Diverse Questions using Variational Autoencoders". In: *CoRR* abs/1704.03493 (2017). arXiv: 1704.03493. URL: http://arxiv.org/abs/1704.03493.

[Koe17] Philipp Koehn. "Neural Machine Translation". In: *CoRR* abs/1709.07809 (2017). arXiv: 1709.07809. URL: http://arxiv.org/abs/1709.07809.

[Aga+18] Basant Agarwal et al. "A deep network model for paraphrase detection in short text messages". In: *Information Processing and Management* (2018), pp. 922–937. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2018.06.005. URL: http://www.sciencedirect.com/science/article/pii/S0306457317308713.

[Pos18] Matt Post. "A Call for Clarity in Reporting BLEU Scores". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191. URL: https://www.aclweb.org/anthology/W18-6319.

[Tak18] Akira Takezawa. *How to implement Seq2Seq LSTM Model in Keras*. 2018. URL: https://towardsdatascience.com/how-to-implement-seq2seq-lstm-model-in-keras-shortcutnlp-6f355f3e5639.

[Xu+18] Qiongkai Xu et al. "D-PAGE: Diverse Paraphrase Generation". In: *CoRR* (2018). arXiv: 1808.04364.

[Zha+18] Chi Zhang et al. "Semantic Sentence Embeddings for Paraphrasing and Text Summarization". In: *CoRR* abs/1809.10267 (2018). arXiv: 1809.10267. URL: http://arxiv.org/abs/1809.10267.

[Che+19] Mingda Chen et al. "Controllable Paraphrase Generation with a Syntactic Exemplar". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019. URL: https://www.aclweb.org/anthology/P19-1599.

[Kaj19] Tomoyuki Kajiwara. "Negative Lexically Constrained Decoding for Paraphrase Generation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019. URL: https://www.aclweb.org/anthology/P19-1607.

[Li+19] Zichao Li et al. "Decomposable Neural Paraphrase Generation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019. URL: https://www.aclweb.org/anthology/P19-1332.

[SR19a] Sarvesh Soni and Kirk Roberts. "A Paraphrase Generation System for EHR Question Answering". In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, 2019. URL: https://www.aclweb.org/anthology/W19-5003.

[SR19b] Sarvesh Soni and Kirk Roberts. "A Paraphrase Generation System for EHR Question Answering". In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, Aug. 2019. URL: https://www.aclweb.org/anthology/W19-5003.

[Zar19] Jaume Zaragoza. *Paraphrasing repository*. 2019. URL: https://github.com/ZJaume/paraphrasing.

[ZSW19] Zhong Zhou, Matthias Sperber, and Alexander Waibel. "Paraphrases as Foreign Languages in Multilingual Neural Machine Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, 2019. URL: https://www.aclweb.org/anthology/P19-2015.

[Com] Association for Computational Linguistics. *Paraphrase Identification (State of the art)*. URL: https://aclweb.org/aclwiki/Paraphrase_Identification_(State_of_the_art).