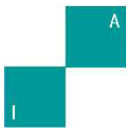
Inteligencia Artificial, 20(59) (2017), 123-139
doi: 10.4114/intartif.vol20iss59pp123-139



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

From Imitation to Prediction, Data Compression vs Recurrent Neural Networks for Natural Language Processing

Juan Andrés Laura, Gabriel Omar Masi, Luis Argerich
Departamento de Computación, Facultad de Ingeniería. Universidad de Buenos Aires
jandreslaura@gmail.com, masigabriel@gmail.com, largerich@fi.uba.ar

Abstract In recent studies [11] [27] [2] Recurrent Neural Networks were used for generative processes and their surprising performance can be explained by their ability to create good predictions. In addition, Data Compression is also based on prediction. What the problem comes down to is whether a data compressor could be used to perform as well as recurrent neural networks in the natural language processing tasks of sentiment analysis and text generation. If this is possible, then the problem comes down to determining if a compression algorithm is even more intelligent than a neural network in such tasks. In our journey, a fundamental difference between a Data Compression Algorithm and Recurrent Neural Networks has been discovered.

Keywords: Natural language processing, Compression algorithms, Neural networks, Predictions.

1 Introduction

One of the most interesting goals of Artificial Intelligence is the simulation of different human creative processes like speech recognition, sentiment analysis, image recognition, automatic text generation, etc. In order to achieve such goals, a program should be able to create a model that reflects how humans think about these problems.

Researchers think that Recurrent Neural Networks (RNN) are capable of understanding the way some tasks are done such as music composition, writing of texts, etc. Moreover, RNNs can be trained for sequence generation by processing real data sequences one step at a time and predicting what comes next [11] [27].

Compression algorithms are also capable of understanding and representing different sequences and that is why the compression of a string could be achieved. However, a compression algorithm might be used not only to compress a string but also to do non-conventional tasks in the same way as neural nets (e.g. a compression algorithm could be used for clustering [5], sequence generation or music composition).

Both neural networks and data compressors should be able to learn from the input data to do the tasks for which they are designed. In this way, someone could argue that a data compressor can be used to generate sequences or a neural network can be used to compress data. In consequence, using the best data compressor to generate sequences should produce better results than the ones obtained by a neural network, otherwise the neural network should compress better than the state of the art in Data Compression.

The hypothesis for this research is that, if compression is based on training from an input data, then the best compressor for a given training set should be able to compete with other algorithms in natural

ISSN: 1137-3601 (print), 1988-3064 (on-line)

language processing tasks. In the present work, this hypothesis will be analyzed for two given scenarios: sentiment analysis and automatic text generation.

2 RNNs for Data Compression

Recurrent Neural Networks and in particular LSTMs were used not only for predictive tasks [8] but also for Data Compression [23]. While the LSTMs were brilliant in their text [27], music [2] and image generation [12] tasks, they were never able to defeat the state of the art algorithms in Data Compression [23].

This might indicate that there is a fundamental difference between Data Compression and Generative Processes and between Data Compression Algorithms and Recurrent Neural Networks. After experiments, a fundamental difference will be shown in this research in order to explain why a RNN can not be the state of the art in Data Compression.

3 Data Compression as an Artificial Intelligence Field

For many authors there is a very strong relationship between Data Compression and Artificial Intelligence [7] [6]. Data Compression is about making good predictions [22] which is also the goal of Machine Learning, a field of Artificial Intelligence.

Essentially, Data Compression involves two important steps: modeling and coding. Coding is a solved problem using arithmetic coding. The difficult task is modeling because it comes down to building a description of the data using the most compact representation; this is again directly related to Artificial Intelligence. Using the Minimal Description Length principle [13] the efficiency of a good Machine Learning algorithm can be measured in terms of how good it is to compress the training data plus the size of the model itself.

A file containing the digits of π can be compressed with a very short program able to generate those digits, gigabytes of information can be compressed into a few thousand bytes. However, the problem arises when trying to find a program capable of understanding that our input file contains the digits of π. In consequence, achieving the best compression rate involves finding a program able to always find the most compact model to represent the data and that is clearly an indication of intelligence, perhaps even of General Artificial Intelligence.

4 Sentiment Analysis

4.1 A Qualitative Approach

Human feelings can be determined according to what they write in many social networks such as Facebook, Twitter, etc.. It looks like an easy task for humans. However, it could be not so easy for a computer to automatically determine the sentiment behind a piece of writing.

The task of guessing the sentiment of texts using a computer is known as Sentiment Analysis and one of the most popular approaches for this task is to use neural networks. In fact, Stanford University created a powerful neural network for sentiment analysis [24] which is used to predict the sentiment of movie reviews taking into account not only the words in isolation but also the order in which they appear. In the first experiment, the Stanford's neural network and a PAQ compressor[1] [19] will be used for sentiment analysis in order to determine whether a user likes or not a given movie (i.e. each movie review will be classified as positive or negative). After that, results obtained will be compared using the percentage of correctly classified movie reviews. Both algorithms will use a public data set [17].

In order to understand how sentiment analysis could be done with a data compressor it is important to comprehend the concept of using Data Compression to compute the distance between two strings using the *Normalized Compression Distance* [16]. The following equation shows how this distance is measured:

[1]PAQ's source code is free and it is available at Mahoney's web [19]

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \quad (1)$$

Where $C(x)$ is the size of applying the best possible compressor to x and $C(xy)$ is the size of applying the best possible compressor to the concatenation of x and y.

The NCD is an approximation to the Kolmogorov distance between two strings using a Compression Algorithm to approximate the complexity of a string because the Kolmogorov Complexity is uncomputable.

The principle behind the NCD is quite simple: when string y is concatenated after string x then y should be highly compressed whenever y is very similar to x because the information in x contains everything needed to describe y. An observation is that $C(xx)$ should be equal, with minimal overhead difference to $C(x)$ because Kolmogorov complexity of a string concatenated to itself is equal to the Kolmogorov complexity of the string.

As introduced, a data compressor performs well when it is capable of understanding the data set that will be compressed. This understanding often grows when the data set becomes bigger and in consequence compression rate improves. However, if the future data (i.e. data that has not been compressed yet) has no relation with compressed data, compression rate would fall. The more similarity the information share, the better compression rate is achieved.

Let $C(X_1, X_2...X_n)$ be a compression algorithm that compresses a set of n files denoted by $X_1, X_2...X_n$. Let $P_1, P_2...P_n$ and $N_1, N_2...N_m$ be a set of n positive reviews and m negative reviews respectively. Then, a review R can be predicted positive or negative using the following inequality:

$$C(P_1, ...P_n, R) - C(P_1, ..., P_n) < C(N_1, ..., N_m, R) - C(N_1, ..., N_m) \quad (2)$$

The formula is a direct derivation from the NCD. When the inequality is not true, a review is predicted negative.

The order in which files are compressed must be considered. As you could see from the proposed formula in 2, the review R is compressed last.

Some people may ask why this inequality works to predict whether a review is positive or negative. So it is important to understand this inequality. Suppose that the review R is a positive one. R will be compressed in order to classify it: if R is compressed after a set of positive reviews then the compression rate should be better than the one obtained if R is compressed after a set of negative reviews because the review R has more related information with the set of positive reviews and in consequence should be compressed better. Interestingly, both the positive and negative set could have different sizes and that is why it is important to subtract the compressed file size of both sets in the inequality. Consider the following example:

> *My favorite movie. What a great story this really was. I'd just like to be able to buy a copy of it but this does not seem possible.*

The previous review has a size of 132 bytes. After compressing the train dataset, the results are:

$C(P_1, ..., P_n, R) - C(P_1, ..., P_n)$: 42 bytes
$C(N_1, ..., N_m, R) - C(N_1, ..., N_m)$: 43 bytes

Given the previous results, the review R is predicted positive because $C(P_1, ..., P_n, R) - C(P_1, ..., P_n)$ is lower than $C(N_1, ..., N_m, R) - C(N_1, ..., N_m)$.

4.2 PAQ for Sentiment Analysis

Using Data Compression for Sentiment Analysis is not a new idea. It has been already proposed in IEEE 12th International Conference [28]. However, the authors did not use PAQ compressor.

PAQ Compressor is taken into account for this research because of its excellent compression rates achieved at Hutter's Prize [1] and many benchmarks such as Matt Mahoney's one [19].

In order to make sentiment analysis of a movie review using PAQ, it is needed to compress each review after compressing both the positive train set and the negative one separately. Given the fact that

compressing each train set takes a considerable time, a checkpoint tool is used in this work. The review is classified positive if the compression rate is better using the positive train set than the one obtained using the negative one. Otherwise, it is classified as negative. If both compression rates are equals, it is classified as inconclusive.

4.3 Data Set Preparation

The Large Movie Review Dataset [17], which has been used for Sentiment Analysis competitions, is used in this research[2].

Table 1: Movie review dataset.

	Positive	Negative
Total	12491	12499
Training	9999	9999
Test	2492	2500

4.4 Experiment Results

In this section, the results obtained are explained by giving a comparison between the data compressor and the Stanford's Neural Network for Sentiment Analysis.

Tables 2, 3 and 4 show the results obtained.

Table 2: PAQ vs RNN. Classification results of the positive reviews.

	Correct	**Incorrect**	**Inconclusive**
PAQ	71.19%	23.72%	5.10%
RNN	46.03%	45.18%	8.79%

Table 3: PAQ vs RNN.Classification results of the negative reviews.

	Correct	**Incorrect**	**Inconclusive**
PAQ	83.20%	13.12%	3.68%
RNN	95.76%	2.08%	2.16%

Table 4: PAQ vs RNN. Overall classification results of the reviews

	Correct	**Incorrect**	**Inconclusive**
PAQ	77.20%	18.41%	4.39%
RNN	70.93%	23.60%	5.47%

Both algorithms have excellent performance when classifying negative reviews, as show in Table 3. In Table 2 are shown the results for positive reviews classification and it can be noticed that results are not as good as the ones obtained with negative reviews. Overall results are shown in Table 4 where you can see that 77.20% of movie reviews were correctly classified by the PAQ Compressor whereas 70.93% were well classified by the Stanford's Neural Network.

There are two main points to highlight according to the result obtained:

1. Sentiment Analysis could be achieved with a PAQ compression algorithm with high accuracy ratio.

[2]Both training set and test set were chosen randomly

2. In this particular case, a higher precision can be achieved using PAQ rather than the Stanford Neural Network for Sentiment Analysis.

It is observed that PAQ can be very accurate to determine whether a review is positive or negative, the miss-classifications were always difficult reviews and in some particular cases the compressor outdid the human label, for example consider the following review:

> *"The piano part was so simple it could have been picked out with one hand while the player whacked away at the gong with the other. This is one of the most bewilderedly trancestate inducing bad movies of the year so far for me."*

This review was labeled positive but PAQ correctly predicted it as negative, since the review is misslabeled it counted as a miss in the automated test.

Analyzers based on words like the Stanford Analyzer tend to have difficulties when the review contains a lot of uncommon words. However, they can work well in longer documents by relying on a few words with strong sentiment like 'awesome' or 'exhilarating' [24]. It was surprising to find that PAQ was able to correctly predict those. Consider the following review:

> *"The author sets out on a "journey of discovery" of his "roots" in the southern tobacco industry because he believes that the (completely and deservedly forgotten) movie "Bright Leaf" is about an ancestor of his. Its not, and he in fact discovers nothing of even mild interest in this absolutely silly and self-indulgent glorified home movie, suitable for screening at (the director's) drunken family reunions but certainly not for commercial - or even non-commercial release. A good reminder of why most independent films are not picked up by major studios - because they are boring, irrelevant and of no interest to anyone but the director and his/her immediate circles. Avoid at all costs!"*

The previous review was classified as positive by the Stanford Analyzer, probably because of words such as "interest, suitable, family, commercial, good, picked", the Compressor however was able to read the real sentiment of the review and predicted a negative label. In cases like this the compressor shows its ability to truly understand data. However, some cases can "hack" both algorithms. In the following example, the review is about an excellent actor that acts as a low-talent comedian. Determining whether it is a positive or negative review is not easy as you can see from the phrases in bold.

> *Chris Rock stars in this remake of Warren Beatty's Heaven Can Wait (itself a remake of the 1941 film Here Comes Mr. Jordan), a comedy about a man who dies before his time, before he can realize his dreams, and his adventures in his new (albeit temporary) body. In the Beatty version, the protagonist was a backup quarterback for the then-Los Angeles Rams. In Rock's hipper version, our lead character is a struggling young -* ***and decidedly low-talent - standup comedian****.

It's very funny to see the razor-sharp Rock* ***playing a bad comedian****. It's kind of like seeing Tom Hanks play a* ***bad actor****. Lance Barton's dream is to play the legendary Apollo Theater on a non-amateur night. But every time he tries out his material, he's booed off the stage lustily - so much so that his nickname becomes "Booie."* ***textHis jokes are lame, his delivery painful****. In short, Lance is everything that the real Chris Rock isn't.
<br /¿Lance is also a bike messenger, and he's riding the streets on his way to try out even more material when BAM! He's hit by a truck. Ok, so maybe he was taken from his body a tenth of a second early by a slightly incompetent angel (Eugene Levy), but hey, he was going to get hit anyway. No dice, it appears Lance isn't due in Heaven until 2044. So what to do? Mr. King (Chazz Palminteri), the "manager" of Heaven, reluctantly agrees to find a new body for the not-quite-dead Mr. Barton. Trouble is, the body they find is of a greedy, old white man. Turns out this fella (a Mr. Wellington) owns all kinds of things - he's the 15th richest man in the country! What luck! You can imagine how Lance will turn*

*things around.

But of course, while in the body of the affluent Mr. Wellington, Lance falls for a gorgeous hospital worker (Regina King). We males know how tough it is to find a female given our own body, but try winning one over while you're an dumpy, old white guy! And it's even worse when she's not impressed by your

* ***This is Rock's first shot at a lead role, and in my opinion he performs admirably. There's still a lot of the standup comedian in him - and, of course, if he ever wants to get diverse roles, he might have to stop incorporating standup routines into the script - but this isn't really a bad thing. Rock's personality - his drive, his delivery, his demeanor, and his passion - are what fuel this film. He's clearly having a lot of fun in the role, and he seems bent on making sure you have fun watching him.***

5 Automatic Text Generation

Recurrent neural networks have been used for automatic text generation [15] [11]. On this tasks, a RNN is trained with texts (or books) in order to sample new characters according to those texts' patterns [3]. Readers may think that the following example was written by Shakespeare but it was not, a RNN was trained with Shakespeare's works and produced it:

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

As a reminder, the ability of good compressors when making predictions is more than evident. It just requires an entry text (i.e. a training set) to be compressed. At compression time, the future symbols will get a probability of occurrence: The higher the probability, the better compression rate for success cases of that prediction, on the contrary, each failure case will take a penalty. At the end of this process, a probability distribution will be associated with that input data [21]. As a result of obtaining this probabilistic model, it will be possible to simulate new samples, in other words, generate text.

In this module, a recurrent neural network and a PAQ compressor will be used to generate text and results will be compared to determine which of them performs better. In the following sections, different scenarios and metrics will be used to make such comparison.

5.1 Data Model

PAQ series compressors use arithmetic coding to encode symbols assigned to a probability distribution. Each probability lies on the interval [0,1) and when it comes to binary coding, there are two possible symbols: 0 and 1.

This compressor uses Models and Contexts, a main part of compression algorithms. Contexts are built from the previous history and can be used to make predictions, for example, the last ten symbols can be linked to compute the prediction of the eleventh. Models process data and assigns probabilities to the future symbols. Moreover, each model's prediction is based on contexts to compute how likely a bit 1 or 0 is next.

PAQ uses an ensamble of several different models. Some of them are based on the previous n characters (or m bits) of processed text, others use whole words as contexts, etc. In order to combine every models'

[3]The source code for this network is available in [15]

prediction, a Model Mixing procedure is included to acquire a complete prediction. So, a neural network will be the mixer to determine the weight of each model [18]:

$$P(1|c)^4 = \sum_{i=1}^{n} P_i(1|c)W_i \tag{3}$$

Where $P(1|c)$ is the probability of bit 1 with context "c",
$P_i(1|c)$ is the probability of bit 1 in context "c" for model i and
W_i is the weight assigned to model i.

In addition, each model adjusts their predictions based on the new information. When compressing, input text is processed bit by bit. On every bit, the compressor updates the context of each model and adjusts the weights as shown in the following equation:

$$W_i = W_i + error_i * \alpha * S_i \tag{4}$$

Given the compressed bit y, the error of each models is defined as
$error_i = y - P_i(1|c)$ and
S_i is a signal that derives from $P_i(1|c)$[5]

The previous equation is important because each W_i must not be adjusted at generation time.

5.2 PAQ for Text Generation

When data set compression is over, PAQ is ready to generate text. A random number is sampled in the [0,1) interval and transformed into a bit 1 or 0 using Inverse Transform Sampling [26]. If the random number falls within the probability range of symbol 1, bit 1 is generated, otherwise, bit 0.

Once that bit is generated, it will be compressed to reset every context for the following prediction. Here, it is essential to update models in a way that if the same context is obtained in two different samples, probabilities must be the same, otherwise it could compute and propagate errors. So, it is necessary to turn off the training process and the weight adjustment of each model at generation time[6].

An example is given in Figure 1, in which PAQ splits the [0,1) interval giving 1/4 of probability to bit 0 and 3/4 of probability to bit 1. When a random number is sampled in this context it is more likely to generate a 1. Each generated bit updates all models' context. However, that bit should not be learned because of its random nature. In other words, PAQ just learns from the training set and then generates random text using that probabilistic model. After 8 bits, a character is generated.

Figure 1: Example of sampling

It was noted that granting too much freedom to the compressor could result in a large accumulation of bad predictions, leading to poor text generation. Therefore, it is proposed to make the text generation more conservative adding a parameter called "temperature" that reduces the possible range of the random number as shown in Figure 2.

On maximum temperature, the random number will be generated in the interval [0,1), giving the compressor maximum degree of freedom to make mistakes, whereas, when the temperature parameter turns minimum, the "random" number will always be 0.5, removing the degree of freedom to commit errors (in this scenario, the highest probability symbol will be generated).

[4]The probability $P(0|c)$ can be interpreted as $1 - P(1|c)$
[5]The signal S_i is often computed as $S_i = stretch(P_i(1|c))$
[6]This is possible because the source code of PAQ is available.

Figure 2: The range is reduced to [0.2, 0.8) when the temperature parameter turns 0.6.

When temperature is around 0.5, the result seems to be actually legible, even if it is not similar to the original text (according to the proposed metrics). This effect is shown at the following randomly generated Harry Potter's snippet:

> *"What happened?" said Harry, and she was standing at him. "He is short, and continued to take the shallows, and the three before he did something to happen again. Harry could hear him. He was shaking his head, and then to the castle, and the golden thread broke; he should have been a back at him, and the common room, and as he should have to the good one that had been conjured her that the top of his wand too before he said and the looking at him, and he was shaking his head and the many of the giants who would be hot and leafy, its flower beds turned into the song.*

5.3 RNN for Text Generation

Previous experiments have been done sampling new text with RNNs [11] [15] [27]]. In this section, a brief explanation of the architecture is given[7].

As suggested in [11], Figure 3 illustrates a basic recurrent neural network prediction architecture. An input vector sequence $x = (x_1, ..., x_T)$ is passed through weighted connections to a stack of N recurrently connected hidden layers to compute first the hidden vector sequences $h^n = (h_1^n, ..., h_T^n)$ and then the output vector sequence $y = (y_1, ..., y_T)$. Each output vector y_t is used to parameterise a predictive distribution $Pr(x_t + 1|y_t)$ over the possible next inputs $x_t + 1$.

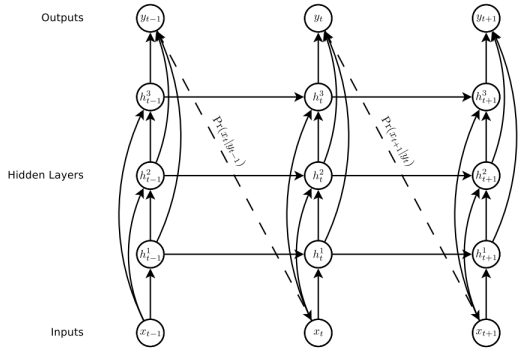


Figure 3: Architechture of a RNN

For this research, a Long Short-Term Memory (LSTM) network has been used. A LSTM is a novel recurrent network architecture in conjunction with an appropiate gradient-based learning algorithm [14].

In order to sample new text, each character of the training text is represented as a vector using 1-of-k encoding (i.e. all zero except for a single one at the index of the character in the vocabulary), and this vector is fed into the RNN. Once the network is trained, a character is fed into it and a you will get a distribution over what characters are likely to come next. Once the distribution is given, a character is

[7]Reader is expected to have knowledge in neural networks

sampled from that distribution and then it is fed right back in to get the next letter [15]. As mentioned in Figure 2, a temperature parameter is also used when sampling text with the RNN used in this research where lower values will give more conservative results whereas using higher values will give more diversity but at cost of more mistakes.

Not only the temperature but also the number of layers and the size of the network are important parameters and they can vary. In fact, varying them is a common practice to find good models. In this research, the network has been trained several times varying such parameters. After each training process, many texts where sampled.

5.4 Metrics

A simple transformation is applied to each text in order to compute metrics. It consists in counting the number of occurrences of each n-gram in the input (i.e. every time a n-gram "WXY..AZ" is detected, it increases its number of occurrences). Then three different metrics were considered:

5.4.1 Pearson's Chi-Squared

How likely it is that any observed difference between the sets arose by chance. The chi-square is computed as:

$$\mathcal{X}^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{5}$$

Where O_i is the observed ith value and E_i is the expected ith value. A value of 0 means equality. This metric is computed using 4-grams.

5.4.2 Total Variation

Each n-gram's observed frequency can be denoted like a probability if it is divided by the sum of all frequencies, P(i) on the real text and Q(i) on the generated one. Total variation distance [9] can be computed according to the following formula:

$$\delta(P, Q) = \frac{1}{2} \sum_{i=1}^{n} |P_i - Q_i| \tag{6}$$

In other words, the total variation distance is the largest possible difference between the probabilities that two probability distributions can assign to the same event. A value of 0 means equality. This metric is computed using 4-grams.

5.4.3 Generalized Jaccard Similarity

It is the size of the intersection divided by the size of the union of the sample sets [4]. Jaccard Similarity is computed using the following formula:

$$J(G, T) = \frac{G \cap T}{G \cup T} \tag{7}$$

Given two non-negative n-dimensional real vectors X, Y , their Jaccard similarity is defined as [4]:

$$J(x, y) = \frac{\sum_{i=1}^{n} min(X_i, Y_i)}{\sum_{i=1}^{n} max(X_i, Y_i)} \tag{8}$$

This last definition is also know as "Weighted Jaccard Similarity". A value of 1 means both texts are equals. This metric is computed using 10-grams.

5.5 About Samples

For each scenario, the RNN was trained several times to find its best hyper-parameters. Asymmetrically, the compressor required to be trained just once. After that, a sampling procedure was executed. It set up different values for "temperature" parameter, allowing these trained models to generate diverse text samples It must be noticed that very high temperatures produce text that is not so similar to the training test, temperatures that are too low aren't also optimal, the best value is usually an intermediate one as shown in Figure 4.

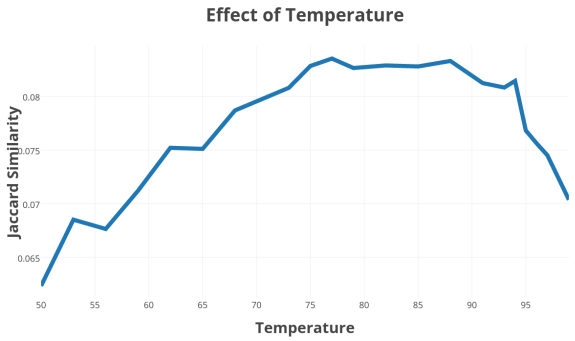


Figure 4: Effect of Temperature in the Jaccard Similarity

In this research, similarity between the input text (e.g. The Complete Works of William Shakespeare) and an automatic generated text sample was measured using a local context and a global one.

1. Local similarity: The input text is splitted into n fragments. Then, the metric is computed using each fragment against the generated text. Given the fact that this technique produces n values, the best value is taken.

2. Global similarity: The metric is computed using the input text against the generated one. There is no fragmentation.

5.6 Results

Turning off the training process and the weights adjustment of each model freezes the compressor's global context at the end of the training set. As a consequence of this event, the last piece of the entry text will be considered as a "big seed".

For example, The King James Version of the Holy Bible includes an index at the end of the text, a bad seed for text generation. Compressing the Bible with that index set an unknown context for PAQ and leaded us to this result:

> *ˆ55And if meat is broken behold I will love for the foresaid shall appear, and heard anguish, and height coming in the face as a brightness is for God shall give thee angels to come fruit.*
>
> *56But whoso shall admonish them were dim born also for the gift before God out the least was in the Spirit into the company*
>
> *[67Blessed shall be loosed in heaven.)*

The index at the end of the file was removed and then PAQ compressed and generated again:

> *ˆ12The flesh which worship him, he of our Lord Jesus Christ be with you most holy faith, Lord, Let not the blood of fire burning our habitation of merciful, and over the whole of life*

with mine own righteousness, shall increased their goods to forgive us our out of the city in the sight of the kings of the wise, and the last, and these in the temple of the blind.

ˆ13For which the like unto the souls to the saints salvation, I saw in the place which when they that be of the bridegroom, and holy partly, and as of the temple of men, so we say a shame for a worshipped his face: I will come from his place, declaring into the glory to the behold a good; and loosed.

ˆ14He that worketh in us, by the Spirit saith unto the earth;and he that they shall not be ashamed before mine old, I come saith unto him the second time, and prayed, saying to flower, and death reigned brass.

The difference is remarkable. Comparing different segments of each input file against each other, it was observed that in some files the last piece was significantly different than the rest of the text. Those unpredictable endings mess up PAQ's generation but it was very interesting to notice that for the RNN did not result in a noticeable difference. This was the first hint that the compressor and the RNN were proceeding in different ways. An example of the impact caused by choosing a seed is given in Figure 5.

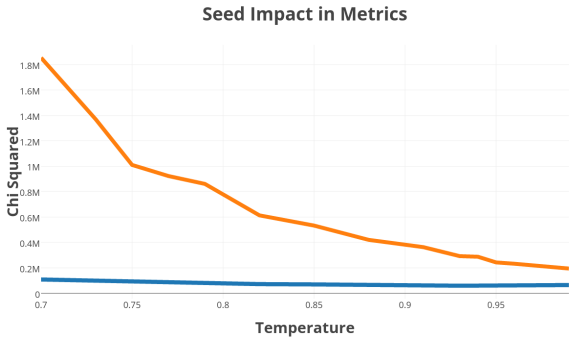


Figure 5: The effect of a chosen seed in the Chi Squared metric. In Orange the metric variation by temperature using a random seed. In Blue the same metric with a chosen one.

Occasionally, the compressor generated text that was surprisingly well written. This is an example of random text generated by PAQ8L after compressing Harry Potter:

CHAPTER THIRTY-SEVEN - THE GOBLET OF LORD VOLDEMORT OF THE FIRE-BOLT MARE!"

Harry looked around. Harry knew exactly who lopsided, looking out parents. They had happened on satin' keep his tables."

Dumbledore stopped their way down in days and after her winged around him.

He was like working, his eyes. He doing you were draped in fear of them to study of your families to kill, that the beetle, he time. Karkaroff looked like this. It was less frightening you.

Another example, this one was generated after compressing Linux Kernel:

```
#ifdef CONFIG_CONSTRUCTORS

struct inode *inode;
   int setup_init(trace->flags, pause_on_oops)
{
   int smp_mb__after_atomic();
```

```
    struct hd_struct *p;
    const char *t;
    int modinfo_next_pid_nr(current));
}
/*
 * Information about the signal context, int that freezers placement states if
 * since periodic);
 */
int cpumask_var_node(&context, unsigned long usermodehelper_execution(struct task_struct
    *prev)
{
    struct module_attribute *attribute, struct module_kobject *mk;
    int ret;
    bool boolval;
}
```

An automatic Game of thrones' snippet:

> *Page 775*
>
> *Summer, or closed with a leather shield, so suddenly. Mero went too. She leaves. There was a fire silver smoke. "No one and losing herself and said the maester gorne a moment." Ser Robin King Joffrey, though for a bolt of the wind and a sword with your brothers and frozen too far and after he was finished the wine and a chainmail. They plotted hands, he had habit for her, slid unknowable,*
>
> *Page 776*
>
> *"From is no tears, and began to gather himself that be well sited and a watch benches can't say, mercy for him to her here. They say me have made more distinct that after another, and seized her. One was many too. His neck and danced into the solar, like the other. He staggered him and something else the sky began, the blankets. The walls son was dry jests. Lord Mormont feebly. I am many horses. Not only was not sorry he had no heed her a moment. I saw his castle, but he thought of his personal emblem. He might have been long and the hall grey mantle opened the outer was never been so heavily he saw no sign of Craven she said of a shadowcat and her smiled looked candle had deflowered, a chunk of bread. "And count. It's not here, eating was a handsome unscathed. Merrett had no one had no more than one faint below into deformed longsword inch ones in the castle, and looked the boots rode over the same silver, for the golden before and brighter from his bedchamber was deeply leather courtesies, and the rest. The red woman was pleased with his wings, of consume us, this is a bill of a single torch, and was a sour on her tongue. It had once the continued until he could close over about a slave chose his true brothers. You know, but the first time my nose to her feet. There was nothing in the walls of it was plain, if defeating into forging from the crown for another. She just another two shoved her own. He has been standing at the dwarf's penny. "You spoke to get a pleasure steward, the sooner. It was when the horses had more screaming and the gods but the look on the causeway the boy chance with the ice where it myself shall, she thought even say her name. No doubt he had gone dark. On around him.*

While the text may not make sense it certainly follows the style, syntax and writing conventions of the training text.

5.7 Metric Results

In this section, the results of both algorithms are shown with the purpose of doing an evaluation of how much similar the results are to the original inputs.

5.7.1 Local similarity

Tables 5, 6 and 7 show local similarity results. It can be noticed that the results obtained using a PAQ compression algorithm are better than the ones obtained by the RNN execpting Poe, Shakespare and Game of Thrones because of a subtle reason that will be explained in the conclusions.

Table 5: Chi Squared Results (Local similarity).

	PAQ8L	**RNN**
Game of Thrones	47790	**44935**
Harry Potter	**46195**	83011
Paulo Coelho	**45821**	86854
Bible	**47833**	52898
Poe	61945	**57022**
Shakespeare	**60585**	84858
Math Collection	**84758**	135798
War and Peace	**46699**	47590
Linux Kernel	**136058**	175293

Table 6: Total Variation %(Local similarity).

	PAQ8L	**RNN**
Game of Thrones	25.21	**24.59**
Harry Potter	**25.58**	37.40
Paulo Coelho	**25.15**	34.80
Bible	**25.15**	25.88
Poe	30.23	**27.88**
Shakespeare	**27.94**	30.71
Math Collection	**31.05**	35.85
War and Peace	**24.63**	25.07
Linux Kernel	**44.74**	45.22

Table 7: Jaccard Similarity (Local similarity).

	PAQ8L	**RNN**
Game of Thrones	0.06118	**0.0638**
Harry Potter	**0.1095**	0.0387
f Paulo Coelho	**0.0825**	0.0367
Bible	**0.1419**	0.1310
Poe	0.0602	**0.0605**
Shakespeare	0.0333	**0.04016**
Math Collection	**0.21**	0.1626
War and Peace	**0.0753**	0.0689
Linux Kernel	**0.0738**	0.0713

5.7.2 Global similarity

The Recurrent Neural Network got better results in global contexts. The results are shown in Tables 8, 9 and 10

Table 8: Chi Squared Results (Global similarity).

	PAQ8L	**RNN**
Game of Thrones	**60541**	62514
Harry Potter	**66008**	363711
Paulo Coelho	**67846**	255951
Bible	838686	**70258**
Poe	99199	**75965**
Shakespeare	180619	**91877**
Math Collection	294999	**100153**
War and Peace	**59625**	62854
Linux Kernel	371226	**198317**

Table 9: Total Variation %(Global similarity).

	PAQ8L	**RNN**
Game of Thrones	21.79	**19.16**
Harry Potter	**25.31**	33.67
Paulo Coelho	**24.92**	30.62
Bible	28.51	**17.21**
Poe	29.63	**21.39**
Shakespeare	29.63	**21.67**
Math Collection	36.46	**22.78**
War and Peace	37.38	**18.81**
Linux Kernel	41.85	**29.70**

Table 10: Jaccard Similarity (Global similarity).

	PAQ8L	**RNN**
Game of Thrones	0.0611	**0.0636**
Harry Potter	**0.0835**	0.0386
Paulo Coelho	**0.0758**	0.0399
Bible	0.0911	**0.1430**
Poe	0.0500	**0.0646**
Shakespeare	0.0332	**0.0401**
Math Collection	0.1351	**0.2094**
War and Peace	0.0427	**0.0761**
Linux Kernel	0.0771	**0.0925**

6 Conclusions

In the sentiment analysis task, an improvement using PAQ over a Neural Network is noticed. A Data Compression algorithm has the intelligence to understand text up to the point of being able to predict its sentiment with similar or better results than the state of the art in sentiment analysis. In some cases the precision improvement was up to 6% which is a lot.

Sentiment analysis is a predictive task, the goal is to predict sentiment based on previously seen samples for both positive and negative sentiment, in this regard a compression algorithm seems to be a better predictor than a RNN.

In the text generation task, the use of a right seed is needed for PAQ algorithm to be able to generate useful text, this was evident in the Bible example. This result is consistent with the sentiment analysis result because the seed is acting like the previously seen reviews, if it is not in sync with the text then the results will not be similar to the original text.

The text generation task showed the critical difference between a Data Compression algorithm and a Recurrent Neural Network and according to our research this is the most important result: Data Compression algorithms are *predictors* while Recurrent Neural Networks are *imitators*.

The text generated by a RNN looks in general better than the text generated by a Data Compressor but if just one paragraph is generated, the Data Compressor is clearly better. PAQ learns from the previously seen text and creates a model that is optimal for predicting what is next, that is why they work so well for Data Compression and that is why they are also very good for Sentiment Analysis or to create a paragraph after seeing the training set.

On the other hand the RNN is a great imitator of what it learned, it can replicate style, syntax and other writing conventions with a surprising level of detail but what the RNN generates is based on the whole text used for training without weighting recent text as more relevant. In this sense, the RNN is better for random text generation while the Compression algorithm should be better for random text extension or completion.

Suppose the text of Romeo & Juliet is located at the end of William Shakespeare's works and then both algorithms use them to generate a sample. As a consequence, PAQ would create a new paragraph of Romeo and Juliet whereas the RNN would generate a Shakespeare-like piece of text. *Data Compressors are better for local predictions and RNNs are better for global predictions.*

This explains why in the text generation process PAQ and the RNN obtained different results for different training tests. PAQ struggled with "Poe" or "Game of Thrones" but was very good with "Coelho" or the Linux Kernel. What really happened was that it was measured how predictable the last piece of the text was!. If the text is very predictable then the best predictor will win, PAQ defeated the RNN by a margin with the Linux Kernel and Paulo Coelho. When the text is not predictable then the ability to imitate in the RNN defeated PAQ. This can be used as a wonderful tool to evaluate the predictability of different authors comparing if the Compressor or the RNN works better to generate similar text. In our experiment it was concluded that Coelho is more Predictable than Poe and it makes all the sense in the world!

As our final conclusion it was shown that Data Compression algorithms show rational behaviour and that they are based on the accurate prediction of what will follow, based on what they have learnt recently. RNNs learn a global model from the training data and can then replicate it. That is why Data Compression algorithms are great **predictors** while Recurrent Neural Networks are great **imitators**. Depending on which ability is needed one or the other may provide the better results.

7 Future Work

From our point of view, Data Compression algorithms could be used with a certain degree of optimality for any Natural Language Processing Task where predictions are made with the recent local context. Completion of text, seed based text generation, sentiment analysis, text clustering are some of the areas where Compressors might play a significant role in the near future.

We have also shown that the difference between a Compressor and a RNN can be used as a way to evaluate the predictability of the writing style of a given text. This might be expended in algorithms that can analyze the level of creativity in text and can be applied to books or movie scripts.

Acknowledgements

This research was supported by Universidad de Buenos Aires. We thank our colleagues who provided insight and expertise that greatly assisted the research.

We thank researchers from Argentine Symposium of Artificial Intelligence for comments that greatly improved the manuscript.

We would also like to show our gratitude to Dr. Matt Mahoney for sharing his knowledge in compression algorithms. We are also immensely grateful to Dr. Alex Graves for his comments on an earlier attempt to make handwriting recognition with PAQ compressor

References

[1] 50'000 prize for compressing human knowledge. http://prize.hutter1.net/.

[2] Oliver Bown and Sebastian Lexer. Continuous-time recurrent neural networks for generative and interactive musical performance. In *Rothlauf F. et al. (eds) Applications of Evolutionary Computing. EvoWorkshops 2006*, volume 3907, pages 652–663, Springer, Berlin, Heidelberg.

[3] Ebru Celikel and Mehmet Emin Dalkilic. Investigating the effects of recency and size of training text on author recognition problem. In *Computer and Information Sciences - ISCIS 2004*, volume 3280, pages 21–30, Springer.

[4] Flavio Chierichetti, Ravi Kumar, Sandeep Pandey, and Sergei Vassilvitskii. Finding the jaccard median. http://theory.stanford.edu/~sergei/papers/soda10-jaccard.pdf.

[5] Rudi Cilibrasi and Paul Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, April 2005.

[6] Ofir David, Shay Moran, and Amir Yehudayoff. On statistical learning via the lens of compression. https://arxiv.org/pdf/1610.03592.pdf, October 2016. arXiv:1610.03592v2.

[7] Arthur Franz. Artificial general intelligence through recursive data compression and grounded reasoning: a position paper. https://arxiv.org/pdf/1506.04366.pdf, June 2015. arXiv:1506.04366v1.

[8] Felix Gers, JÃ$\frac{1}{4}$rgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. https://pdfs.semanticscholar.org/1154/0131eae85b2e11d53df7f1360eeb6476e7f4.pdf, January 1999.

[9] Alison L. Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International Statistical Review*, 70:419–435, September 2002.

[10] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385. Springer, 2012.

[11] Alex Graves. Generating sequences with recurrent neural networks. https://arxiv.org/pdf/1308.0850.pdf, June 2014. arXiv:1308.0850v5.

[12] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. https://arxiv.org/pdf/1502.04623.pdf, February 2015. arXiv:1502.04623v2.

[13] Peter Grunwald. A tutorial introduction to the minimum description length principle. https://arxiv.org/pdf/math/0406077.pdf, June 2004. arXiv:math/0406077v1.

[14] Sepp Hochreiter and JÃ$\frac{1}{4}$rgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[15] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, May 2015.

[16] Ming Li and Paul Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.

[17] Andrew Maas, Raymond Daly, Peter Pham, Dan Huang, Andrew Ng, and Christopher Potts. Learning word vectors for sentiment analysis. http://ai.stanford.edu/~ang/papers/acl11-WordVectorsSentimentAnalysis.pdf.

[18] Matt Mahoney. Fast text compression with neural networks. https://cs.fit.edu/~mmahoney/compression/mmahoney00.pdf.

[19] Matt Mahoney. The paq data compression series. http://mattmahoney.net/dc/paq.html.

[20] Matt Mahoney. Adaptive weighing of context models for lossless data compression. https://cs.fit.edu/~mmahoney/compression/cs200516.pdf, 2005.

[21] Matt Mahoney. Data compression explained. http://mattmahoney.net/dc/dce.html#Section_4, April 2013.

[22] Joel Ratsaby. Prediction by compression. https://arxiv.org/pdf/1008.5078.pdf, August 2010. arXiv:1008.5078v1.

[23] JÃ$\frac{1}{4}$rgen Schmidhuber and Stefan Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7:142–146, January 1996.

[24] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf, 2013.

[25] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf, 2013.

[26] Mark Steyvers. *Computational Statistics with Matlab*. May 2011.

[27] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf, 2011.

[28] Dominique Ziegelmayer and Rainer Schrader. Sentiment polarity classification using statistical data compression models. *IEEE 12th International Conference on*, December 2012.