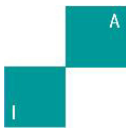
Inteligencia Artificial, 20(59) (2017), 123-132
doi: 10.4114/intartif.vol20iss59pp123-132



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

Departemento de Computación, Facultad de Ingeniería. Universidad de Buenos Aires
please@do.not.include.email

jandreslaura@gmail.com, masigabriel@gmail.com, largerich@fi.uba.ar

Abstract In recent studies [9] [22] [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 automatic 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 a Recurrent Neural Network has been discovered.

Keywords: KeyWord1, Keyword2, KeyWord3.

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 [9] [22].

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 [4], 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 but if this is not true then the neural network should compress better than the state of the art in data compression.

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

The hypothesis for this research is that, if compression is based on learning from the input data set, then the best compressor for a given data set should be able to compete with other algorithms in natural language processing tasks. In the present work, this hypothesis will be analyzed for two given scenarios: sentiment analysis and automatic text generation.

2 Data Compression as an Artificial Intelligence Field

For many authors there is a very strong relationship between Data Compression and Artificial Intelligence [6] [5]. Data Compression is about making good predictions [18] 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 compression. 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 [11] 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.

3 RNNs for Data Compression

Recurrent Neural Networks and in particular LSTMs were used not only for predictive tasks [7] but also for Data Compression [19]. While the LSTMs were brilliant in their text [22], music [2] and image generation [10] tasks, they were never able to defeat the state of the art algorithms in Data Compression [19].

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

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 text 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 [20] 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 neural network and a PAQ compressor[1] [16] will be used for doing sentiment analysis of movie reviews 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 for movie reviews [14].

[1] PAQ's source code is free and it is available at Mahoney's web [16]

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* [13].

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

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, it is not true when future data (i.e. data that has not been compressed yet) has no relation with already compressed data because the more similar the information it is 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:

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

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, 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.

4.2 Data Set Preparation

The Large Movie Review Dataset [14], 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

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

4.3 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 [23]. 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 [16].

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 using the positive train set is better than the one obtained using the negative train set. Otherwise, it is classified as negative. If both compression rates are equals, it is classified as inconclusive.

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.

The following table shows the results obtained

Table 2: PAQ vs RNN Classification Results.

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

As you could see from the previous table, 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. In this particular case, a higher precision can be achieved using PAQ rather than the Stanford Neural Network for Sentiment Analysis.

We observed that PAQ was very accurate to determine whether a review was positive or negative, the missclassifications 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' [20]. 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!"

This 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 the ability to truly understand data.

5 Automatic Text Generation

This module?s goal is to automatically generate text with a PAQ series compressor and compare it with Andrej Karpathy RNN?s results [12], using specifics metrics and scenarios.

The ability of good compressors when making predictions is more than evident. It just requires an entry text (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 [17]. As a result of this probabilistic model, it will be possible to simulate new samples, in other words, generate text.

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. Moreover, this compressor uses contexts, a main part of compression algorithms. They 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.

PAQ uses an ensamble of several different models to compute how likely a bit 1 or 0 is next. Some of these models are based on the previous n characters of m bits of seen text, other models use whole words as contexts, etc. In order to weight the prediction performed by each model, a neural network is used to determine the weight of each model [15]:

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

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 of the neural network.

Generally, as you compress more information, predictions improve.

5.2 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 [21]. In other words, 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 obteined in two different samples, probabilities must be the same, otherwise it could compute and propagate errors. Seeing that, it was necessary to turn off the training process and the weight adjustment of each model at generation time. This was also possible because the source code of PAQ is available.

It was noted that granting too much freedom to our 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.

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).

When temperature is around 0.5 the results are very legible even if they are not as similar to the original text (according to the proposed metrics). It can be seen 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 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 ?WXYZ? is detected, it increases its number of occurrences). Then three different metrics were considered:

5.3.1 Pearson's Chi-Squared

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

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

Where O_i is the observed ith value and E_i is the expected ith value. A value of 0 means equality.

5.3.2 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 [8] can be computed according to the following formula:

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

Figure 1: 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 becaused 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.

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.

5.3.3 Generalized Jaccard Similarity

It is the size of the intersection divided by the size of the union of the sample sets [3]. Jaccard Similarity is computed using the following formula:

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

A value of 1 means both texts are equals.

6 References

References should be clear and complete. Sample references are these [**?**] [**?**] [**?**] and this one [**?**]. Do not forget to include DOI numbers in your references when available. You can associate hyperlinks to DOI numbers in LaTeXusing the \doi command, see for example [**?**]. Please, pay attention to the difference with [**?**] that explicitly includes a URL, and not a DOI reference. See below the format of references. LaTeXusers are encouraged to use Bibtex.

6.1 What is a DOI?

The DOI (Digital Object Identifier) is a unique identifier assigned by publishers to electronically published items (books, journal papers, etc.) The DOI system provides a framework for persistent identification. The physical location of a digital object may change over time, but its DOI name will not change. Currently, a DOI number can be resolved prefixing http://dx.doi.org/ to the DOI. Therefore links of the form http://dx.doi.org/[DOI] are expected to work properly over time, regardless of the physical location changes of the referenced items. To learn more about the DOI see `http://www.doi.org`

6.2 Where can I find the DOI of my references?

DOI numbers should appear explicitly in the first page of printed journal papers, either at the top or bottom of the page, like in figure 2. Electronic journals display DOI numbers in the papers access or home pages, like in figure 3. If you cannot find the DOI of a reference at first sight it probably does not have one, so do not worry. Many articles do not have DOI numbers assigned (specially older ones).

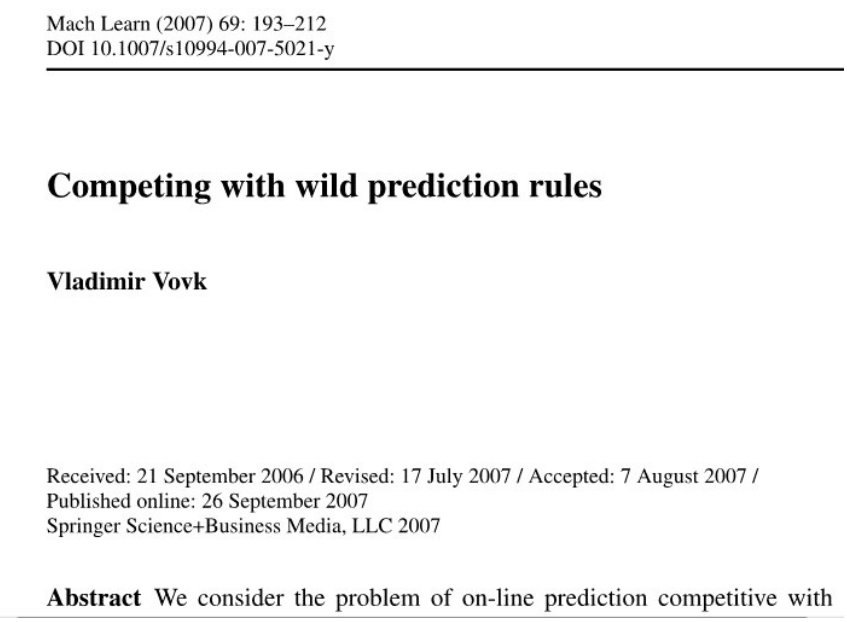
6.3 Why is it important to include DOI numbers in references (when available)?

New papers published by Inteligencia Artificial are assigned a DOI number as a free service to their authors. Documents receiving a DOI are expected to include DOI numbers in their own references.

Including DOI numbers in your references (when available) will make your paper more useful and comfortable to read for readers and reviewers. If you are a LaTeXuser and include DOI numbers as explained above, the related hyperlinks will be added automatically to your document.

7 Journal Scope

Inteligencia Artificial is a quarterly journal promoted and sponsored by Iberamia (Sociedad Iberoamericana de Inteligencia Artificial)(http://iberamia.dsic.upv.es/inicial.html). The journal publishes high-quality original research papers reporting theoretical or applied advances in all branches of Artificial Intelligence. In addition to rapid publication and dissemination of unsolicited contributions, Inteligencia Artificial is committed to producing monographs and special issues on topics of special relevance to the AI community.



Mach Learn (2007) 69: 193–212
DOI 10.1007/s10994-007-5021-y

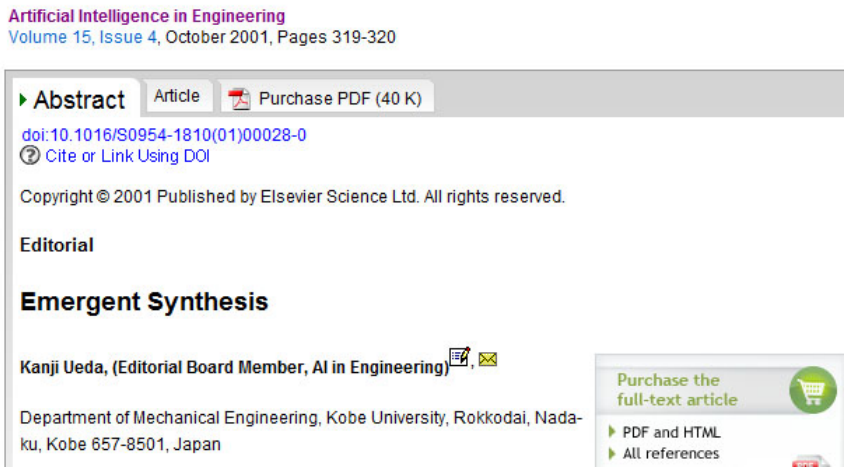
Competing with wild prediction rules

Vladimir Vovk

Received: 21 September 2006 / Revised: 17 July 2007 / Accepted: 7 August 2007 /
Published online: 26 September 2007

Abstract We consider the problem of on-line prediction competitive with

Figure 2: DOI in a printed document.



Artificial Intelligence in Engineering
Volume 15, Issue 4, October 2001, Pages 319-320

Abstract Article Purchase PDF (40 K)

doi:10.1016/S0954-1810(01)00028-0
Cite or Link Using DOI

Editorial

Emergent Synthesis

Kanji Ueda, (Editorial Board Member, AI in Engineering)

Department of Mechanical Engineering, Kobe University, Rokkodai, Nada-ku, Kobe 657-8501, Japan

Figure 3: DOI in a document available on the Web.

8 Manuscript Submission and Review

Paper submission should be carried out through the journal's Web page http://journal.iberamia.org. Click the "ABOUT" button and follow the author's guidelines.

Following this process the author implicitly transfers copyright of accepted papers to IBERAMIA, accepting their publication electronically (freely available through the Journal's Web pages) and in printed volumes.

Inteligencia Artificial welcomes research papers written in Spanish or English. Manuscripts should be uploaded in PDF format through the journal's Web pages (http://journal.iberamia.org - link 'Submissions'). All research papers falling within the journal's scope will be subject to double-blind peer-review by at least two anonymous reviewers. Authors must omit their names in submitted manuscripts. The journal publishes four issues per year, each one at the start of each season (spring, summer, autumn, and winter). The journal will not carry out editing of manuscripts. Authors should adhere strictly to the norms in the "Journal's formatting guidelines". Accepted papers will be published electronically at the journal's Web pages, with free access from the Internet. The journal encourages authors to add links in their personal pages pointing to the journal's pages containing their contributions.

9 Originality and Copyright

Research papers submitted to Inteligencia Artificial cannot have been published previously, nor be under review for publication elsewhere. However, the journal welcomes extended and improved versions of papers published in conferences and workshops, which will be subject to a new peer-review process. Authors implicitly accept publication of submitted papers by , both electronically (with free access from the Internet) and in print, as well as dissemination through other media and entities with which IBERAMIA may establish agreement, transferring IBERAMIA all the necessary rights.

10 Monographs and Special Issues

The journal welcomes monograph or special issue proposals related to specific areas of Artificial Intelligence. Proposals to coordinate monographs or special issues should be sent through email to the Editor (editor@iberamia.org). Contributions to monographs or special issues can be original works including new research papers, extended and improved versions of papers published in conferences and workshops, and survey papers that summarize the topics of a particular monograph or special issue. In all cases, the publication of contributions is conditioned to acceptance in a new peer-review process carried out by an international scientific committee under the journal's supervision.

11 Theses Summaries

Inteligencia Artificial is specially interested in the publication of Ph.D. dissertation summaries that provide a quick and state-of-the-art reference in particular topics of Artificial Intelligence. Summaries can be submitted following the regular submission process, stating "RESUMEN DE TESIS" in the heading, and should be limited to 2-4 pages.

Acknowledgements

This is the place for acknowledgements.

References

[1] 50'000 prize for compressing human knowledge.

[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] Flavio Chierichetti, Ravi Kumar, Sandeep Pandey, and Sergei Vassilvitskii. Finding the jaccard median.

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

[5] Ofir David, Shay Moran, and Amir Yehudayoff. On statistical learning via the lens of compression, October 2016. arXiv:1610.03592v2.

[6] Arthur Franz. Artificial general intelligence through recursive data compression and grounded reasoning: a position paper, June 2015. arXiv:1506.04366v1.

[7] Felix Gers, JÃ$\frac{1}{4}$rgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm, January 1999.

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

[9] Alex Graves. Generating sequences with recurrent neural networks, June 2014. arXiv:1308.0850v5.

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

[11] Peter Grunwald. A tutorial introduction to the minimum description length principle, June 2004. arXiv:math/0406077v1.

[12] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015.

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

[14] Andrew Maas, Raymond Daly, Peter Pham, Dan Huang, Andrew Ng, and Christopher Potts. Learning word vectors for sentiment analysis.

[15] Matt Mahoney. Fast text compression with neural networks.

[16] Matt Mahoney. The paq data compression series.

[17] Matt Mahoney. Data compression explained, April 2013.

[18] Joel Ratsaby. Prediction by compression, August 2010. arXiv:1008.5078v1.

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

[20] 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, 2013.

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

[22] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks, 2011.

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