2019

Multi-Element Long Distance Dependencies: Using SPk Languages to Explore the Characteristics of Long-Distance Dependencies

Abhijit Mahalunkar
Technological University Dublin, abhijit.mahalunkar@tudublin.ie

John Kelleher
Technological University Dublin, john.kelleher@tudublin.ie

Recommended Citation

Mahalunkar, A. & Kelleher, J.D. (2019) Multi-Element Long Distance Dependencies: Using SPk Languages to Explore the Characteristics of Long-Distance Dependencies, *Workshop on Deep Learning and Formal Languages: Building Bridges,* Florence, Italy, August, 2nd 2019. DOI: 10.18653/v1/W19-3904

Multi-Element Long Distance Dependencies: Using SP*k* Languages to Explore the Characteristics of Long-Distance Dependencies

Abhijit Mahalunkar
Applied Intelligence Research Center
Technological University Dublin
Dublin, Ireland
abhijit.mahalunkar@mydit.ie

John D. Kelleher
ADAPT Research Center
Technological University Dublin
Dublin, Ireland
john.d.kelleher@dit.ie

Abstract

In order to successfully model Long Distance Dependencies (LDDs) it is necessary to understand the full-range of the characteristics of the LDDs exhibited in a target dataset. In this paper, we use Strictly *k*-Piecewise languages to generate datasets with various properties. We then compute the characteristics of the LDDs in these datasets using mutual information and analyze the impact of factors such as (i) *k*, (ii) length of LDDs, (iii) vocabulary size, (iv) forbidden subsequences, and (v) dataset size. This analysis reveal that the number of interacting elements in a dependency is an important characteristic of LDDs. This leads us to the challenge of modelling multi-element long-distance dependencies. Our results suggest that attention mechanisms in neural networks may aide in modeling datasets with multi-element long-distance dependencies. However, we conclude that there is a need to develop more efficient attention mechanisms to address this issue.

1 Introduction

Long Distance Dependencies (LDDs) describe an interaction between two (or more) elements in a sequence that are separated by an arbitrary number of positions. LDDs are related to the rate of decay of statistical dependence of two points with increasing time interval or spatial distance between them. For example, in English there is a requirement for subjects and verbs to agree, compare: "*The **dog** in that house **is** aggressive*" with "*The **dogs** in that house **are** aggressive*". This dependence can be computed using information theoretic measure i.e. *Mutual Information* (Cover and Thomas, 1991; Paninski, 2003; Bouma, 2009; Lin and Tegmark, 2017).

To date most research on LDDs has focused on the distance the dependency spans within the sequence. However, as our analysis will show the complexity of LDDs not only arises from the distance but also a number of other factors, including: (i) the number of unique symbols in a dataset, (ii) the size of the dataset, (iii) the number of interacting symbols within an LDD, and (iv) the distance between the interacting symbols. In this paper we use SPk languages to explore the complexity of LDDs. The motivation for using the SP*k* language modelling task, is that the standard sequential benchmark datasets provide little to no control over the factors which directly contribute to LDD characteristics. By contrast, using SP*k* languages we can generate benchmark datasets with varying degrees of LDD complexity by modifying the grammar of the SP*k* language (Rogers et al., 2010; Fu et al., 2011; Avcu et al., 2017).

One aspect of LDDs that has been neglected in the research on LDDs is the complexity that arises from a multi-element dependency (i.e., dependencies that involves interactions between more than 2 elements). By controlling *k* in the SP*k* grammar, it is possible to generate datasets with varying degrees of multi-element dependency. This multi-element dependencies pose specific challenges to neural architectures that may require these architectures to be augmented with pointer or attention mechanisms. We explore whether attention mechanism can help with multi-element LDDs using two models, Transformer-XL (Dai* et al., 2019) and AWD-LSTM (Merity et al., 2018). Transformer-XL employs multi-head attention mechanism along with recurrence mechanism. Whereas, AWD-LSTM is a weight dropped LSTM which does not employ any attention/pointer mechanism.

The Transformer-XL and AWD-LSTM models are both *language models*. A language model accepts a sequence of symbols and predicts the next symbol in the sequence. The accuracy of a language model is dependent on the capacity of the

Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges, pages 34–43
Florence, Italy, August, 2nd 2019.

model to capture the LDDs in the data on which it is evaluated. The standard evaluation metric for language models is *perplexity*. Perplexity is the measurement of the confusion or uncertainty of a language model as it predicts the next symbol in a sequence, and the lower the perplexity of a model the better the performance of the model.

2 Related Work: Neural Networks and Artificial Grammars

Formal Language Theory, primarily developed to study the computational basis of human language is now being used extensively to analyze any rule-governed system (Chomsky, 1956, 1959; Fitch and Friederici, 2012). Formal languages have previously been used to train RNNs and investigate their inner workings. The Reber grammar (Reber, 1967) was used to train various 1st order RNNs (Casey, 1996; Smith and Zipser, 1989). The Reber grammar was also used as a benchmarking dataset for LSTM models (Hochreiter and Schmidhuber, 1997). Regular languages, studied by Tomita (Tomita, 1982), were used to train 2nd order RNNs to learn grammatical structures of the strings (Watrous and Kuhn, 1991; Giles et al., 1992).

Regular languages are the simplest grammars (type-3 grammars) within the Chomsky hierarchy which are driven by regular expressions. For neural network research an interesting subclass of regular languages is the Strictly k-Piecewise languages. Strictly k-Piecewise languages are natural and can express some of the kinds of LDDs found in natural languages (Jager and Rogers, 2012; Heinz and Rogers, 2010). This presents an opportunity of using SPk grammar to generate benchmarking datasets (Avcu et al., 2017; Mahalunkar and Kelleher, 2018). In Avcu et al. (2017), LSTM networks were trained to recognize valid strings generated using SP*2*, SP*4*, SP*8* grammar. LSTM could recognize valid strings generated using SP*2* and SP*4* grammars but struggled to recognize strings generated using SP*8* grammar, exposing the performance bottleneck of LSTM networks. It has also been observed that the performance of LSTMs on SP*2* datasets degraded when the length of the LDDs in the datasets were increased, this was done by increasing the maximum length of the generated strings of SP*2* (Mahalunkar and Kelleher, 2018).

3 Preliminaries

3.1 Strictly k-Piecewise Languages (SPk)

SPk languages form a subclass of regular languages. Subregular languages can be identified by mechanisms much less complicated than Finite-State Automata. Many aspects of human language such as local and non-local dependencies are similar to subregular languages (Jager and Rogers, 2012). More importantly, there are certain types of long distance (non-local) dependencies in human language which allow finite-state characterization (Heinz and Rogers, 2010). These type of LDDs can easily be characterized by SPk languages and can be easily extended to other processes.

A language L, is described by a finite set of unique symbols Σ and Σ^* (*free monoid*) is a set of finite sequences or strings of zero or more elements from Σ.

Example 3.1. Consider, $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ where $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are the unique symbols. A *free monoid* over Σ contains all concatenations of these unique symbols. Thus, $\Sigma^* = \{\lambda, \sigma_1, \sigma_1\sigma_2, \sigma_1\sigma_3, \sigma_1\sigma_4, \sigma_3\sigma_2, \sigma_3\sigma_1\sigma_3, \sigma_2\sigma_1\sigma_4\sigma_3, \ldots\}$.

Definition 3.1. Let, u denote a string, e.g. $u = \sigma_3\sigma_2$. The length of a string u is denoted by $|u|$, and if $u = \sigma_3\sigma_2$ then $|u|=2$. A string with length zero is denoted by λ.

Definition 3.2. A string v is a *subsequence* of string w, iff $v = \sigma_1\sigma_2 \ldots \sigma_n$ and $w \in \Sigma^*\sigma_1\Sigma^*\sigma_2\Sigma^* \ldots \Sigma^*\sigma_n\Sigma^*$, where $\sigma \in \Sigma$. A *subsequence* of length k is called a *k-subsequence*. Let $\text{subseq}_k(w)$ denote the set of subsequences of w up to length k.

Example 3.2. Consider, $\Sigma = \{a, b, c, d\}$, $w = [acbd]$, $u = [bd]$, $v = [acd]$ and $x = [db]$. String u is a *subsequence* of length $k = 2$ or *2-subsequence* of w. String v is a *3-subsequence* of w. However, string x is *not a subsequence* of w as it does not contain $[db]$ subsequence.

SPk languages are defined by grammar G_{SPk} as a set of permissible *k-subsequences*. Here, k indicates the number of elements in a dependency. Datasets generated to simulate 2 elements in a dependency will be generated using SP*2*. This is the simplest dependency structure. There are more complex chained-dependency structures which require higher k grammars.

Example 3.3. Consider L, where $\Sigma = \{a, b, c, d\}$. Let G_{SP2} be SPk grammar which is comprised of permissible *2-subsequences*. Thus, $G_{SP2} = \{aa,$

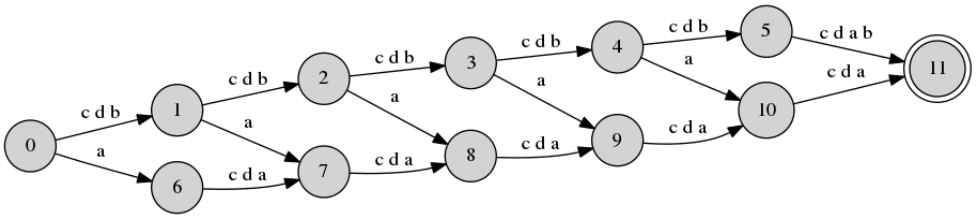

Figure 1: The automaton for G_{SP2} where n_l=6

ac, ad, ba, bb, bc, bd, ca, cb, cc, cd, da, db, dc, dd}. G_{SP2} grammar is employed to generate SP2 datasets.

Definition 3.3. Subsequences which are not in the grammar G are called *forbidden subsequences*[1].

Example 3.4. Consider Example 3.3, although {*ab*} is a possible *2-subsequence*, it is not part of the grammar G_{SP2}. Hence, {*ab*} is a *forbidden subsequence*.

Example 3.5. Consider strings *u, v, w*: *u* = [*bbcbdd*], *v* = [*bbdbbbcbddaa*] and *w* = [*bbabbbcbdd*], where $|u|$ = 6, $|v|$ = 12 and $|w|$ = 10. Strings *u* and *v* are valid SP*2* strings because they are composed of subsequences that are in G_{SP2}. However, *w* is an invalid SP*2* string because *w* contains {*ab*} a subsequence which is a *forbidden subsequence*. These constraints apply for any string *x* where $|x| \in \mathbb{Z}$.

Example 3.6. Let G_{SP3} = {*aaa, aab, abb, baa, bab, bba, bbb, ...*} and *forbidden subsequence* = {*aba*} be an SP*3* grammar which is comprised of permissible 3-*subsequences*. Thus, *u* = [*aaaaaaab*], where $|u|$ = 8 is a valid SP*3* string and *v* = [*aaaaabaab*], where $|v|$ = 9 is an invalid SP*3* string as defined by the grammar G_{SP3}.

The maximum extent of LDD exhibited by a certain SP*k* language is equal to the length of the strings generated which abide by the grammar. However, as per definition 3.2, the strings generated using this method will also exhibit dependencies of shorter lengths. It should be noted that the length of the LDD is not the same as *k*. The length of the LDD is the maximum distance between two elements in a dependency, whereas *k* specifies the number of elements in the dependency (as defined in the the SP*k* grammar).

Example 3.7. As per Example 3.5, *v* = [*bbdbbbcbddaa*], consider *b* in the first position, *subsequence* {*ba*} exhibits dependency of 10 and 11. Similarly, *subsequence* {*bd*} exhibits dependency of 2, 9, 10, etc.

Figure 1 depicts a finite-state diagram of G_{SP2}, which generates strings of synthetic data. Consider a string *x* from this data, ∀ generated strings *x* generated using grammar G_{SP2}: $|x|$ = 6. The *forbidden subsequence* for this grammar is {*ab*}. Since {*ab*} is a *forbidden subsequence*, the state diagram has no path (from state 0 to state 11) because such a path would permit the generation of strings with {*ab*} as a subsequence, *e.g.* {*abcccc*} Traversing the state diagram generates valid strings *e.g.* {*accdda, caaaaa*}.

Various G_{SPk} could be used to define an SP*k* depending on the set of *forbidden subsequences* chosen. Thus, we can construct rich datasets with different properties for any SP*k* language. *forbidden subsequences* allow for the elimination of certain logically possible sequences while simulating a real world dataset where the probability of occurrence of that particular sequence is highly unlikely. Every SP*k* grammar is defined with at least one *forbidden subsequence*.

3.2 Plotting LDD Characteristics

Mutual information has previously been used to analyse LDDs in datasets. For example, in Ebeling and Poeschel (2002), mutual information was used to analyse the maximum length of the LDDs in two English literary texts, Moby Dick by H. Melville and Grimm's tales. Another example, is the work of Lin and Tegmark (2017) who analyzed the LDD characteristics in *enwik8* dataset.

Mutual information measures dependence between random variables X and Y. These random variables have marginal distributions $p(x)$ and $p(y)$ and are jointly distributed as $p(x, y)$ (Cover and Thomas, 1991; Li, 1990). Mutual information, $I(X;Y)$ is defined as;

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (1)$$

[1]Refer section *5.2. Finding the shortest forbidden subsequences* in (Fu et al., 2011) for method to compute *forbidden sequences* for SP*k* language

If X and Y are not correlated, in other words if they are independent to each other, then $p(x)p(y) = p(x, y)$ and $I(X;Y) = 0$. However, if X and Y are fully dependent on each other, then $p(x) = p(y) = p(x, y)$ which results in the maximum value of $I(X;Y)$.

Mutual information can also be expressed using the *entropy* of X and Y i.e. $H(X)$, $H(Y)$ and their *joint entropy*, $H(X, Y)$ as given in the equations below:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \quad (2)$$

$$H(X) = -\sum_{x} p(x) \log p(x) \quad (3)$$

In our algorithm, we compute the $H(X)$ using Grassberger's corrections (Grassberger, 2003).

$$H(X) = \log N - 1/N \sum_{i=1}^{k} N_i \psi(N_i) \quad (4)$$

where N_i is the frequency of unique symbol i, $N = \sum N_i$, K is the number of unique symbols, and $\psi(N_i)$ is the *logarithmic derivative of the gamma function* of N_i.

In order to measure dependence between any two symbols at a distance D in a sequence, we design random variables X and Y so that X holds the subsequence of the original sequence from index 0 till $|dataset| - 1 - D$, and Y holds the subsequence from index D till $|dataset| - 1$; where D represents spacing between the symbols and $|dataset|$ or LEN is the size of the dataset. Next we define a random variable XY that contains a sequence of paired symbols one from X and one from Y, where the symbols in a pair have the same index in X and Y. Algorithm 1 explains the details.

Using this information, and Equations 2 and 4, we calculate the mutual information $I(X, Y)$ at a distance D in a sequence. We define the *LDD characteristics* of any given sequential dataset as a function of mutual information $I(X;Y)$ over the distance D. Once we have calculated the mutual information within a dataset at the different distances D which range between 1 to $|dataset|$ we can then plot the LDD characteristics as a graph of distance D versus mutual information at D. The LDD characteristics are plotted on a *log-log axis*, the *x-axis* defines the distance between a pair of symbols and the *y-axis* marks the mutual information.

Algorithm 1 Computing LDD Characteristics

```
for D ← 1, |dataset| do
    X ← dataset[0 : |dataset| − D]
    Y ← dataset[D : |dataset|]
    XY ← zero-matrix of size (K^X, K^Y)
    for i ← 0, |X| do
        Increment XY[X[i], Y[i]]
    end for
    Compute N_i^X, N^X, K^X for X
    Compute N_i^Y, N^Y, K^Y for Y
    Compute N_i^XY, N^XY, K^XY for XY
    Compute H(X), H(Y) and H(X, Y) Eq. 4
    I[D] ← H(X) + H(Y) − H(X, Y)
end for
```

4 LDD characteristics of SP*k* datasets

Natural datasets present little to no control over the factors that affect LDDs. This, limits our ability to understand LDDs in more detail. SP*k* languages exhibit some types of LDDs occurring in natural datasets. Moreover, by modifying the SP*k* grammar we can control the LDD characteristics within a dataset generated by the grammar. To understand and validate the interaction between an SP*k* grammar and the characteristics of the data it generates, we used a number of datasets of SP*k* grammar and analyzed the properties of these datasets. Every dataset is a collection of strings and these strings strictly follow the grammar. Hence the size of the dataset ($|dataset|$) is the sum of the size of all the strings. The datasets were generated using *foma* (Hulden, 2009) and *python* (Avcu et al., 2017; Mahalunkar and Kelleher, 2018)[2]. Below we analyze the impact of various factors on the resulting LDD characteristics.

4.1 Impact of *k*

A dependency may arise within a dataset due to two or more interacting elements. A two element dependency is the simplest dependency structure and is analyzed by models addressing LDDs. However, multiple element dependency may not be rare and if not modeled correctly may well contribute to the errors of a model. Lack of knowledge of these dependency structures within benchmark datasets present significant limitation in comparing the performance of different models aimed at addressing LDDs. Different model architectures

[2]The scripts and details of these datasets are available at https://github.com/silentknight/DelFol-ACL-2019

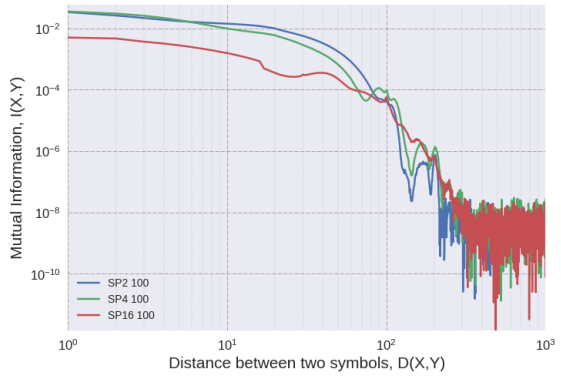

Figure 2: LDD characteristics of datasets of SP*2*, SP*4* and SP*16* grammar exhibiting LDD of length 100.

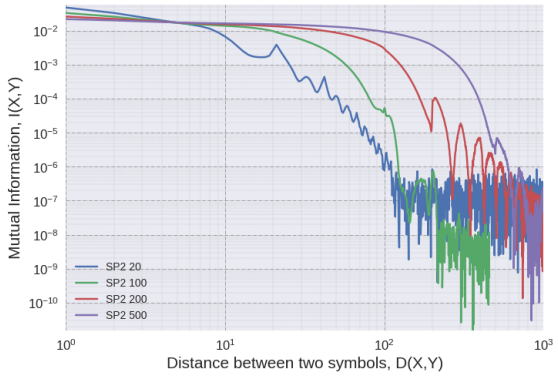

Figure 3: LDD characteristics of datasets of SP*2* grammar exhibiting LDDs of length 20, 100, 200 and 500.

may be able to represent one type of LDD characteristic to a greater extent than another (e.g., distance versus k). However, unless the experimenter is able to control the LDD characteristics present in a dataset it is not possible to disentangle which characteristics a given model struggles with based solely on the models performance on the data. SPk grammar addresses this problem by providing control over both dependency distance and k.

We use SP*2*, SP*4* and SP*16* grammars to generate a set of datasets. With k=$\{2, 4, 16\}$, we generate datasets with different dependency structures with interacting elements $2, 4$, and 16 respectively. Figure 2 plots the LDD characteristics of SP*2*, SP*4* and SP*16* grammars. They contain uniform distribution of strings of string length l where $60 \leq l \leq 100$. The maximum length of strings in each of these datasets is 100. Hence, we can observe steeper mutual information decay beyond $D > 100$. k defines the number of correlated or dependent elements in a dependency rule. As k increases the grammar becomes more complex and there is an overall reduction in frequency of the dependent elements in a given sequence (due to lower probability of these elements occurring in a given sequence). Hence, the mutual information is lower. This can be seen with dataset of SP*16* as compared to SP*2* and SP*4*. It is worth noting that datasets with lower mutual information curves tend to present more difficulty during modeling (Mahalunkar and Kelleher, 2018).

4.2 Impact of LDD length

The distance or length between two interacting elements present significant challenge in modeling LDDs as the model is required to store the context of the interacting element persistently. The success of a model is dependent on whether it is capable of storing the required length of the contexts as dictated by the dataset.

We generated strings of maximum length 20 $(2 \leq l \leq 20)$, 100 $(21 \leq l \leq 100)$, 200 $(101 \leq l \leq 200)$ and 500 $(201 \leq l \leq 500)$ using SP*2* grammar. As explained in Example 3.6, by increasing the length of the generated strings, the distance between dependent elements is also increased, resulting in longer LDDs. Consequently, using this string lengths we can simulate LDD lengths of $20, 100, 200$ and 500.

Figure 3 plots LDD characteristics of SP*2* languages with maximum string length of $20, 100, 200, 500$. The point where mutual information decay is faster, the *inflection point*, lies around the same point on *x-axis* as the maximum length of the LDD. This confirms that SPk can generate datasets with varying lengths of LDDs.

4.3 Impact of Vocabulary Size

We analyze the impact of vocabulary size on LDD characteristics, we generate SP2 grammars where Σ_1=$\{a, b, c, d\}$ $(V$=$4)$ and Σ_2=$\{a, b, c, d, ..., x, y, z\}$ $(V$=$26)$, where V is vocabulary size. The impact of vocabulary size can be seen in figure 4. Both these datasets contain strings of maximum length 20. Hence the mutual information decays at 20. Both curves have identical decay indicating a similar grammar. However the overall mutual information of the dataset with V=26 is much lower then the mutual information of the dataset with V=4. This is because a smaller vocabulary results in an increase in the probability of the occurrence of each individual elements.

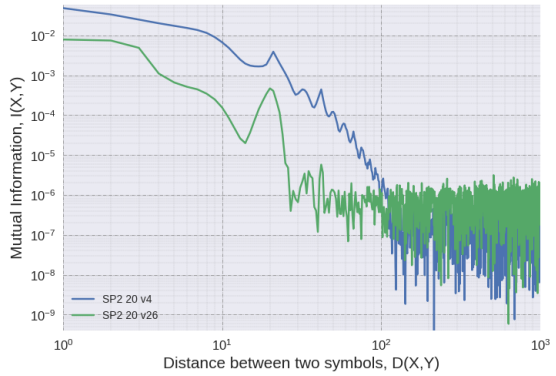

Figure 4: LDD characteristics of datasets of SP2 grammar with vocabulary of 4 and 26.

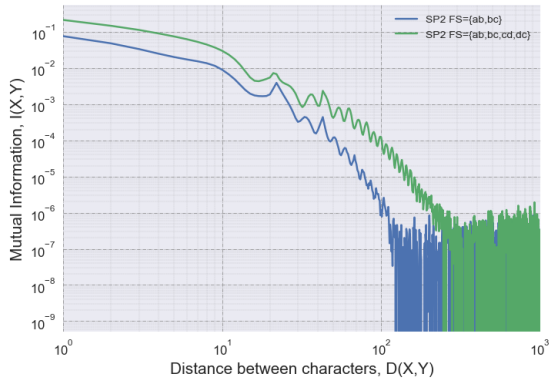

Figure 5: LDD characteristics of datasets of SP2 grammar with varying forbidden subsequences.

4.4 Impact of *forbidden subsequences*

forbidden subsequences control the complexity of a given grammar. We choose two sets of forbidden subsequences for SP2 grammar, $\{ab, bc\}$ and $\{ab, bc, cd, dc\}$.

Figure 5 plots the LDD characteristics of SP2 grammar with two set of *forbidden subsequences* as $\{ab, bc\}$ and $\{ab, bc, cd, dc\}$. It is seen that the dataset with more forbidden subsequences exhibited mutual information decay tending towards a power law decay as compared to an exponential decay by dataset with less forbidden subsequences. As explained in Lin and Tegmark (2017), datasets with exponential decay tend to exhibit Markovian behavior and thus are easy to model as compared to datasets with power law decay. Complex LDDs in a dataset result in power law decay. Thus, by controlling the forbidden subsequences, one can introduce more complex LDDs.

4.5 Impact of dataset size

Another factor to analyze is the impact of the size of the dataset ($|dataset|$) on LDDs of the same

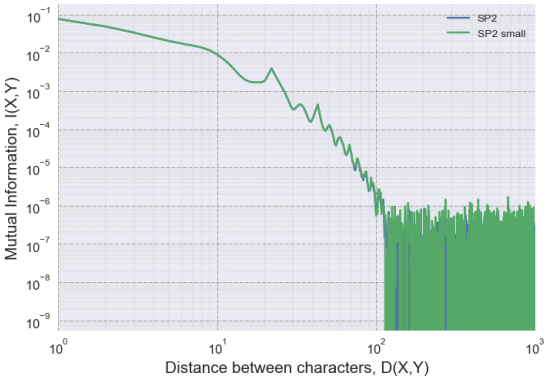

Figure 6: LDD characteristics of datasets of SP2 grammar with varying size of the datasets

grammar. We generate two sizes of the same SP2 grammar to study the impact of the size of the data on the LDD characteristics, where one dataset is twice the size of the other.

In figure 6 we can observe that LDD characteristics of datasets sampled from the same grammar are less likely to be affected by the size of the dataset.

5 Multi-Element Long Distance Dependencies: Attention Mechanisms and *k*

As discussed above in section 4.1, LDDs may arise due to multiple interacting elements, which can be referred to as multi-element long-distance dependency (ME-LDD). Current LDD research primarily focuses on developing models which are capable of retaining contextual information across long distances in sequential datasets. This approach, which focuses solely on dependency distance, may be insufficient in addressing the problems arising due to ME-LDDs. However, recent advances in attention mechanisms and memory networks may be able to represent ME-LDDs. In this section we investigate two models, Transformer-XL (Dai* et al., 2019) (with attention mechanism) and AWD-LSTM (ASGD Weight-Dropped LSTM) (without attention mechanism) (Merity et al., 2018) so as to analyze how attention mechanisms help in modeling datasets with ME-LDDs.

The Transformer-XL model augments vanilla Transformer models by introducing a recurrence mechanism to the Transformer architecture. This recurrence effectively encodes an arbitrarily long context into a fixed size representation over constrained memory and computation. A vanilla Transformer is made up of Multi-Head Attention

Models	Test Perplexity in *bpc*			
	SP2	SP4	SP8	SP16
1	1.6855	1.8038	1.9611	2.0759
2	1.413	1.486	1.658	1.708

Table 1: Perplexity score of 1: Transformer-XL and 2: AWD-LSTM models of SP2, SP4, SP8 and SP16 datasets with vocabulary size V=4

Models	Test Perplexity in *bpc*			
	SP2	SP4	SP6	SP8
1	4.6846	4.7320	4.7384	4.7385
2	4.525	4.635	4.707	4.708

Table 2: Perplexity score of 1: Transformer-XL and 2: AWD-LSTM models of SP2, SP4, SP6 and SP8 datasets with vocabulary size V=26

and Feed Forward layers which aides in adding positional information to the embedded representation. The other model we tested, the AWD-LSTM, uses a weight-drop mechanism so as to aid in regularization of the LSTM network. Hence this model does not explicitly uses attention mechanism.

We trained both these models with variants of SPk grammar where k=$\{2, 4, 8, 16\}$ for vocabulary size V=4 and k=$\{2, 4, 6, 8\}$ for vocabulary size V=26. Hence, there are in total 8 datasets. Every dataset contains uniform distribution of strings with string length l where $60 \leq l \leq 100$. The string length ordering is not maintained so as to not bias the models. In-order to train the language models, every dataset is split into training/validation/test sets. All the training sets for vocabulary size V=4 contain ≈195000 number of strings and the size of the training set is $\approx 24MB$. Similarly, all the training sets for vocabulary size V=26 contain ≈222000 number of strings with size of $\approx 40MB$. All the test and validation sets contain ≈16000 strings with size of $2MB$[3]. The hyperparameters for both the models were reused from the *Penn Treebank character* model (Dai* et al., 2019; Merity et al., 2018).

Table 1 lists the test perplexity scores of both the models in *bits per character* for datasets with vocabulary size V=4 and table 2 lists the test perplexity scores of both the models for datasets with vocabulary size V=26. We plot the test perplexity scores of all the datasets as function of k in

[3]The datasets and scripts used for training the models can be found at https://github.com/silentknight/DelFol-ACL-2019

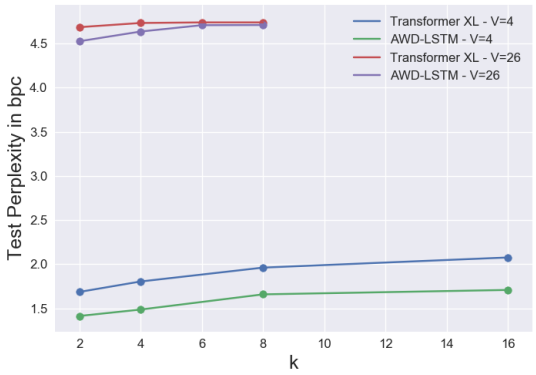

Figure 7: Test perplexity score of Transformer-XL and AWD-LSTM models of SP2, SP4, SP8 and SP16 datasets.

figure 7. Here we observe two distinct sets of perplexity growth curves. It can be seen that, as the size of the vocabulary changes, the test perplexity score across all the models also changes relatively. This confirms that the vocabulary size of the dataset does impact the perplexity score of the models. This is attributed to the lower mutual information in the LDD characteristics as explained in section 4.3.

Switching our focus to the growth in perplexity as k increases, we tried to understand how the presence of attention mechanism impacts the ability of the neural architectures to model the ME-LDDs as k increases. For datasets with V=26, the impact of attention mechanism in Transformer-XL is apparent. The test perplexity score remains almost unchanged which we attribute to the presence of an attention mechanism. However, AWD-LSTM model struggles to maintain test perplexity scores as k increases, for datasets with V=26 (higher vocabulary size).

For datasets with V=4, it can be seen that the test perplexity of Transformer-XL model as a function of k scales exponentially. The presence of exponential relation indicates that the attention mechanism of the Transformer helps the model as k increases in the SPk grammar: the exponential relationship indicates that the growth in model's perplexity appears to saturate as k increases. For AWD-LSTM model, the test perplexity with k also increases at similar rate as that of Transformer-XL. This could be attributed to smaller vocabulary size, which leads to less complex dependency structure even at higher value of k. Such datasets could be easily modeled using

non-attention mechanisms as seen in the figure 7.

Comparing the test perplexity scores in figure 7 for both of these models, it can be observed that the overall test perplexity score of Transformer-XL model for across all the datasets as compared with AWD-LSTM model is higher. This could be attributed to over-parameterization of the Transformer-XL model. Transformer-XL uses $\approx$44 million parameters as compared to $\approx$18 million used by AWD-LSTM model to model these sub-regular languages. It should also be noted that the height of the LDD characteristics of all the datasets is significantly lower than natural datasets (Mahalunkar and Kelleher, 2018). Consequently the mutual information is very small at a given distance D. This leads to higher perplexity in language models modeling them.

We also observed no apparent impact on the value of test perplexity scores of all the models on training sets with twice the number of strings. This can be substantiated by observing the LDD characteristics in section 4.5. Limitation of *foma* tool to generate datasets with various properties prevented us from exploring more complex datasets. *E.g.* we were unable to generate SP16 dataset with vocabulary size of V=26 due to *stack full* error.

6 Discussion

The LDD characteristics of a dataset is indicative of the presence of a certain type of grammar in the dataset. Our experiments reveal that even though a specific grammar does induce similar LDD characteristics, there are subtle variations. These variations depend on a number of factors such as size of the vocabulary, length of contextual relations, dependency structure (for e.g. "*k*" and "*forbidden subsequences*"). This analysis improves our understanding of the complex nature of the LDDs. This analysis can be extended to natural datasets in an effort to better understand the datasets. Thus, if a sequential model such as recurrent neural architecture intends to model a dataset, knowing these factors would greatly benefit in selecting the best hyper-parameters of the sequential model. By training Transformer-XL and AWD-LSTM model with datasets possessing various properties, it was possible to observe the impact of various properties on the perplexity score. Also, the impact of multi-head attention mechanism on the vocabulary size is quite evident. Our results suggest that the Transformer-XL performs much better with increase in vocabulary size. It is also evidenced by its SoTA results on *WikiText103* dataset (V=267735) (Dai* et al., 2019).

7 Conclusion

The majority of neural network research on sequential models focuses solely on modelling dependencies across long distances. However, the dependencies that occur in sequential data can also be multi-element. Furthermore, the vocabulary size, and the forbidden subsequences within a grammar also contribute to the difficulty of modelling the dependencies within a dataset.

In natural datasets all of these factors interact, and can confound the analysis for model performance. However, using SP*k* languages it is possible to synthesize sequential datasets and control the type of dependencies exhibited in these datasets. Using a mutual information based analysis of SP*k* synthesized datasets we examined how the different language characteristics (vocabulary size, forbidden subsequences) and dependency characteristics (length, *k*) are reflected in the datasets generated by SP*k* grammars. Furthermore, our results suggest that attention mechanisms in neural networks may aide in modeling datasets with multi-element long-distance dependencies. Although we encourage developing more efficient models.

The potential impact of this work for neural networks research include: an appreciation of the multifaceted nature of LDDs; a procedure for measuring LDD characteristics within a dataset; an understanding of how different hyper-parameters setting on an SP*k* based dataset synthesis process (string length, *k*, forbidden subsequences, vocabulary size) affect the mutual information profile of the resulting dataset.

Acknowledgment

This research was partly supported by the ADAPT Research Centre, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds. The research was also supported by an IBM Shared University Research Award. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU under NVIDIA GPU Grant used for this research.

References

Enes Avcu, Chihiro Shibata, and Jeffrey Heinz. 2017. Subregular complexity and deep learning. In *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML)*.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction.

M. Casey. 1996. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA.

Zihang Dai*, Zhilin Yang*, Yiming Yang, William W. Cohen, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Language modeling with longer-term dependency.

Werner Ebeling and Thorsten Poeschel. 2002. Entropy and Long range correlations in literary English. 26(2):241–246.

W Tecumseh Fitch and Angela D Friederici. 2012. Artificial grammar learning meets formal language theory: an overview. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1933–1955.

Jie Fu, Jeffrey Heinz, and Herbert G. Tanner. 2011. An algebraic characterization of strictly piecewise languages. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6648 LNCS:252–263.

C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, and Y. C. Lee. 1992. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405.

P. Grassberger. 2003. Entropy Estimates from Insufficient Samplings. *ArXiv Physics e-prints*.

Jeffrey Heinz and James Rogers. 2010. Estimating strictly piecewise distributions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 886–896, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32. Association for Computational Linguistics.

G. Jager and J. Rogers. 2012. Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970.

Wentian Li. 1990. Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60(5):823–837.

Henry W. Lin and Max Tegmark. 2017. Critical behavior in physics and probabilistic formal languages. *Entropy*, 19(7).

Abhijit Mahalunkar and John D. Kelleher. 2018. Understanding Recurrent Neural Architectures by Analyzing and Synthesizing Long Distance Dependencies in Benchmark Sequential Datasets. *arXiv e-prints*, page arXiv:1810.02966.

Abhijit Mahalunkar and John D. Kelleher. 2018. Using regular languages to explore the representational capacity of recurrent neural architectures. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 189–198, Cham. Springer International Publishing.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253.

Arthur S. Reber. 1967. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6(6):855–863.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On Languages Piecewise Testable in the Strict Sense. In *The Mathematics of Language*, pages 255–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

A. W. Smith and D. Zipser. 1989. Encoding sequential structure: experience with the real-time recurrent learning algorithm. In *International 1989 Joint Conference on Neural Networks*, pages 645–648 vol.1.

Masaru Tomita. 1982. Learning of construction of finite automata from examples using hill-climbing : RR : Regular set Recognizer. *Proceedings of Fourth International Cognitive Science Conference*, pages 105–108.

Raymond L. Watrous and Gary M. Kuhn. 1991. Induction of finite-state automata using second-order recurrent networks. In *NIPS*.