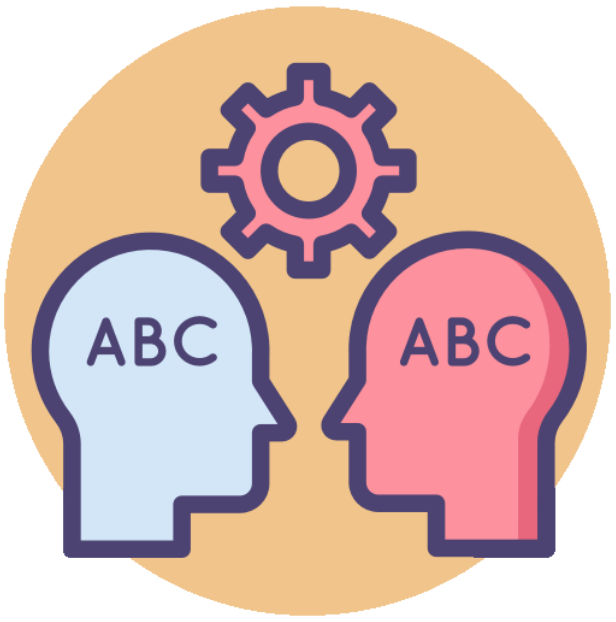
The effect of Zipf's law on the learnability of language



Max N. Bongers

Layout: typeset by the author using LaTeX.
Cover illustration: https://www.flaticon.com/natural-language-processing

The effect of Zipf's law on the learnability of language

Max N. Bongers 11349875

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors
Dhr. dr. W.H. Zuidema
Dhr. V. Vogelmann

Institute for Logic, Language and Computation
Cognition, Language and Computation lab
University of Amsterdam
Science Park 107
1098 XG Amsterdam

June 25th, 2021

Abstract

How learnable is language as a distribution? And how do the properties of this distribution affect its learnability? These are natural and important questions in statistics and machine learning, since they are fundamental to choosing models and forming expectations about their performance. Statistical theory would make us expect that most of its distributional properties make learning language particularly difficult. A prominent example of such a property is Zipf's law which asserts that the word distribution in language follows roughly a power law and which is the main culprit for the massive sparsity in language that makes language modelling such a complex task.

Learnability is also the subject of long and intense debates in cognitive linguistics but here, to the contrary, researchers have recently begun to hypothesise that distributional aspects of language such as Zipf's law exist precisely because they make communication systems easier to learn (Christiansen and Chater 2008). In this thesis several LSTMs were trained on a large set of corpora, these sets all differed in their conformity to Zipf's law. The perplexities of each of these models were then measured and compared. The results show that a lower presence of Zipf's law reduces the learnability of a language.

Contents

1. Introduction

The word distribution that is present in language can be seen as an imbalanced class problem. Some words occur frequently and some words occur extremely infrequently. class imbalance has been studied thoroughly as it relates to machine learning (Johnson and Khoshgoftaar 2019; Chawla 2009). In general imbalanced data can add difficulty for models, most learners will be biased towards the majority class, and sometimes even ignore the minority class entirely. However the class imbalance as it relates to NLP is under-explored. There have been countless studies made on Zipf's law, mostly focused on where the law originates from (Cancho and Solé 2003; W. Li 1992). In spite of this only a few small scale experiments have been done on the effect of the law on learnability (Kurumada, Meylan, and Frank 2011). However without a way of acquiring non-Zipfian corpora it is a difficult process to measure this effect. Fortunately, recently a new method was created which subsamples natural corpora and creates new corpora in which the influence of Zipf's law is reduced (Vogelmann, Cornelissen, and Zuidema 2020).

Using this method it is now more feasible to research the effects of Zipf's law on language. The research question for this paper is thus twofold: "Does Zipf's law affect the learnability of language", and if so, "How does Zipf's law affect the learnability of language?". Based on the previous research on both the origins and the learnability of Zipf's law, the hypothesis is that Zipf's law does affect the learnability of language, and it affects it in a positive way (Zipf 1949; Kurumada, Meylan, and Frank 2011). Learnability is not something that can be easily defined, and thus also not easily measured. In this thesis language models have been trained on corpora, the models's perplexities serve as surrogates of learnability. These models were also used to generate text, which was analysed. Additionally minimum compression size was used as an approximation of learnability.

The contribution of this thesis is to bring first evidence from a large-scale experiment on the effect of Zipf's law on learnability to the table. To do this, corpora differing in their conformities to Zipf's law, have been compared to eachother on these previously named measures.

2. Zipf's law

The distribution over words used in languages follow a relatively simple power law; when ranking all the words used in a (large enough) natural corpus by their frequencies, the frequency of a word will be inversely proportional to its rank. In simple terms this means that the most common word will occur twice as often as the second most common, three times as often as the third most common and so on. This simple mathematical phenomenon is known as Zipf's law (Zipf 1949).

$$f(r) \propto \frac{1}{r^{\alpha}}$$

Where f is the frequency of a word and r is the rank. The actual frequencies are dependent on the size of the corpus. An equation that better resembles the word distribution was proposed by (Mandelbrot 1953). This is done by introducing a parameter β that is added to the rank.

$$f(r) \propto \frac{1}{(r+\beta)^{\alpha}}$$

For $\alpha \approx 1$ and $\beta \approx 2.7$. When testing the 'Zipfianness' of corpora and when speaking of Zipf's law in general in this thesis, this is the equation that will be used. The result of this equation on a large enough corpus is a straight line on a log-log plot.

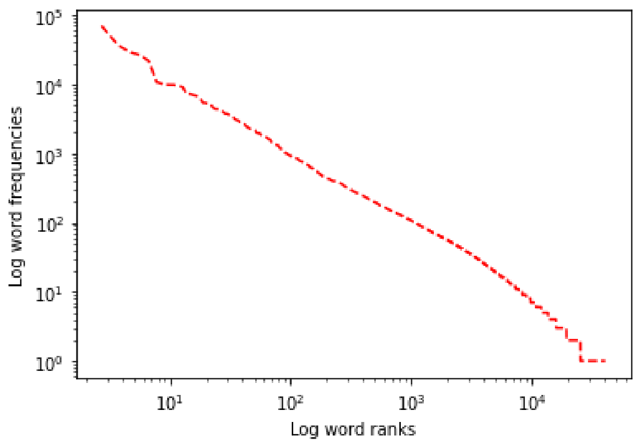


Figure 2.1: Logarithmic plot of Zipf's law on the Brown corpus

To emphasize the scarcity this creates in languages; 20% of unique words account for 80% of all used words (Newman 2005). This can better be seen in a normal plot. Note the long tail of low frequency words which is a key property of Zipf's law, and the main reason for the existing sparsity of words in language. Words such as 'the', 'of' and 'a' will make up the high frequency words, and the low frequency words will be more specific words such as 'regeneration'.

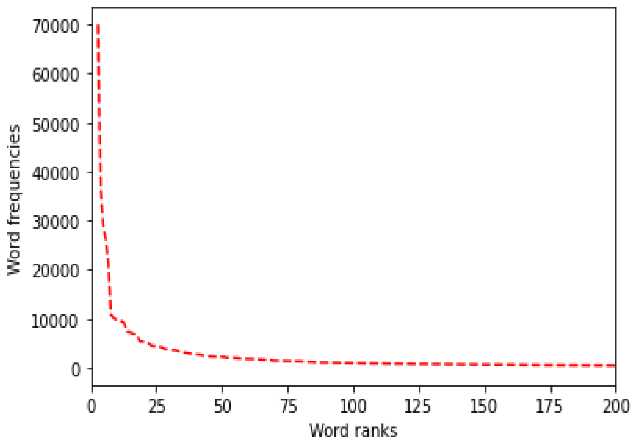


Figure 2.2: Zipf's law on the Brown corpus

2.1 Piantadosi's method

One problem with the derivation of the law is not addressed by the previous equation is that the rank of a word is based on its frequency in the same corpus, this means that when plotting the frequencies against the ranks, these are always highly correlated. When plotting the Zipf line, this will then have correlated errors between the frequency and rank axis of the plot. This problem can be avoided by splitting the corpus in two and calculating the frequencies of one corpus, and the ranks of the other (Piantadosi 2014). Words with zero occurrences after splitting the corpus are not used.

In figure 2.3 Zipf's law is plotted with split corpora, where in a two-dimensional histogram words fall in frequency/rank space. Each bin (hexagon) is shaded from blue to white on a logarithmic scale depending on how many words fall into the bin. The red line is the estimated fit of Zipf's law using a maximum likelihood estimation (MLE).

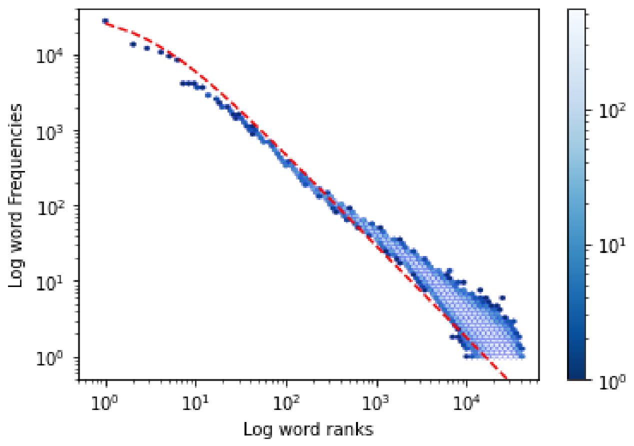


Figure 2.3: Piantadosi Zipf's law on the Brown corpus

As can be seen the log-log plot is linear. The plot shows high variance at lower frequency words, this is to be expected because there is more noise at lower frequency words and thus more uncertainty about the shape of the curve. The MLE also follows the word distribution well.

2.2 Previous research

Over the past century there have many hypotheses about why such a complicated thing as language follows such a relatively simple power law. George Zipf himself hypothesised that it was a consequence of the principle of least effort (Zipf 1949). Speakers will prefer to convey their thoughts in as few words as possible, while listeners, to understand what was being said, preferred larger, more specific vocabularies. This 'clash' then results in a compromise of word usage, that is the cause of the distribution we now see in languages. There are many more possible explanations that have been hypothesised about, to this day the origins of the law are up for debate and, if the past is any indication, will continue to be so in the future.

A few researchers have studied the effect of Zipf's law on learnability before. (Kurumada, Meylan, and Frank 2011) is an experiment on the effect of Zipf's law on learnability. Humans were shown several (made up) languages with different word distributions, ranging from Zipfian to uniform. They found that languages with Zipfian distributions were easier to learn for people than uniform ones. Especially performance on unknown words was better, suggesting that higher frequency words such as 'the' and 'a' surround low frequency words and help with word segmentation. This study was very small however, the languages were only made up of 6-36 word types. Another study (Hendrickson and Perfors 2019) was done which found similar results; people performed better on Zipfian distributions than on uniform ones. However (Blythe, K. Smith, and A. D. Smith 2010) found that the sparsity of words in natural language makes it difficult to learn the meanings of infrequent words. The few studies that have been done so far do not confidently point in one direction or another.

2.3 Measuring Zipfianness

2.3.1 Zipfianness

Being Zipfian or non-Zipfian is not a binary property (Vogelmann, Cornelissen, and Zuidema 2020), by definition all languages have a monotonically decreasing rank-frequency relationship, this is simply an artefact of how the rank is calculated. This means that all languages's rank-frequency relationships are somewhere between a language with a vocabulary of only one word, and one where all words are equally frequent. A Zipfian language will then be somewhere in the middle, between these two extremes. An important assumption that has been made in (ibid.), in which a big part of the research of this thesis was sketched, is that language is unbounded. This assumption is in part made due to research done that shows that the amount of low frequency words increase with corpus size (Blevins, Milin, and Ramscar 2017). Because of this the researches argue that human language should be assumed to have an unbounded vocabulary (the amount of unique words), at least theoretically. Every corpus can then be seen as just a sample of that infinite language distribution.

2.3.2 Entropy

To measure the Zipfianness of a corpus, we first need to measure the entropy of Zipf's law. Entropy is the average of expected length in bits (binary digits) that it takes to encode the suprisal of a word. The entropy of a distribution is the expected amount of information in an event drawn from that distribution (Cover 1999). Formally entropy is defined as

$$H(P) = -\sum_{x \in X} P(x) \log_2 P(x)$$

Where P is the distribution over the vocabulary X. $H(P)$ is then the length in bits that it takes to encode the suprisal about every word in the vocabulary.

Knowning that an unlikely event has happened is more informative than knowning that a likely event has happened. In other words, if a word has a low probability according to the distribution, and thus a high suprisal, it will take more bits to encode this than if the opposite occurs. In the most extreme case, where an outcome is certain, the entropy will be 0, because there is no suprisal. On the other end, where the entropy will be at its highest, is in the case where the distribution is completely uniform, the entropy will then be the logarithm of the length of the vocabulary.

The distribution of Zipf's law over a vocabulary W is given by

$$P_{\alpha,\beta}(w) = \frac{(r(w)+\beta)^{-\alpha}}{\zeta(\alpha,\beta)}$$

Where ζ is the Hurwitz zeta function (Berndt 1972). which is formally defined as

$$\zeta(\alpha,\beta) = \sum_{n=1}^{\infty}(i+\beta)^{-\alpha}$$

The entropy of the distribution over Zipf's law is given by (for a full derivation of this formula see (Vogelmann, Cornelissen, and Zuidema 2020))

$$H(P_{\alpha,\beta}) = \frac{-\alpha * (\frac{\partial}{\partial\alpha}\zeta(\alpha,\beta))}{\zeta(\alpha,\beta)} + \log(\zeta(\alpha,\beta))$$

2.3.3 Typicality

The Zipfianness of a corpus can now be measured by calculating its typicality w.r.t. Zipf's law. In other words, how typical it is to obtain corpus C when sampling from $P_{\alpha,\beta}$ (the distribution over Zipf's law). The typicality of a corpus C w.r.t. $P_{\alpha,\beta}$ is formally defined as (ibid.)

$$a(C^n; P_{\alpha,\beta}) = H(P_{\alpha,\beta}) - \frac{1}{n}\sum_{i=1}^{n}\log_2\frac{1}{P_{\alpha,\beta}(w_i)}$$

where C^n is the corpus, n is the amount of words in the corpus, $H(P_{\alpha,\beta})$ is the entropy of Zipf's law, and $P_{\alpha,\beta}(w_i)$ is the probability of a word from the corpus C^n occurring under the probability distribution of Zipf's law. This means that the more likely the distribution from the corpus C^n is to arise under the probability distribution $P_{\alpha,\beta}$, the lower the typicality will be (I.E. the corpus C^n is more typical w.r.t. $P_{\alpha,\beta}$). If C^n is sampled from $P_{\alpha,\beta}$, and with a large enough sample size, the typicality will converge to 0. Conversely, if C^n is not sampled from $P_{\alpha,\beta}$, the typicality will converge to the Kullback-Leibler divergence between the two distributions (Kullback and Leibler 1951). This now gives us a way to measure the Zipfianness of a corpus by their typicality as measured by $(C^n,\ P_{\alpha,\beta})$. To reiterate, when speaking of typicality in this thesis, a higher value will mean that the corpus is less typical w.r.t. Zipf's law, I.E. it will be less like human languages.

3. Perplexity

The models that have been trained return a probability distribution, given input, the model will return the chance of occurrence for each word in the models vocabulary. A models perplexity then, is the measurement of how well this probability distribution predicts words given unseen data (Jurafsky and C. Manning 2012). The formula that is used to calculate this is as follows

$$b^{\frac{1}{N}} = \sum_{i=1}^{N} \log_b q(x_i)$$

Where b is a constant, N is the length of the test set, q is the model that returns a probability distribution and x_i is the probability of the actual next token. A good model will ascribe higher probabilities to the actual token in the test set, and in turn, as the name suggests, be less 'perplexed' by the token. This means that a lower perplexity means a better performing model. Perplexity is generally a better measure of the performance of a model then say, accuracy, since there are so many different words in the vocabulary, meaning that the model has many different tokens it can predict, this makes it so accuracy on the test set will usually be low. Furthermore, text is often ambiguous with its meaning, for instance consider the following sentence:

- I'm going to the store to buy some tupperware.

Now consider truncating the word 'tupperware' off.

- I'm going to the store to buy some

There are countless words that it could possibly be, the chance that the model would specifically predict 'tupperware' as a token is very low. There is simply not enough context to confidently say, and if it did predict it correctly, the question remains if it is even a smart prediction, since tupperware is not a very common item to get from a store. That is why perplexity is a better measure of performance, the word 'tupperware' in this context should not have as small of a probability as a verb, but not as high of a probability as bread, for instance. Perplexity doesn't look at the predicted token, but rather at what probability the actual token has in the distribution returned by the model. This means that even if the model is wrong about its most probable token, it can still get a decent performance score for at least assigning a reasonable probability to the actual token.

4. Algorithm and parameter choices

Neural networks are powerful machine learning models that achieve excellent results on numerous different tasks. A problem that arises when using a neural network on sequential data such as language is that they don't remember things that they learn. More specifically, after every epoch during training the network doesn't remember what it learned during the previous iteration, except the updated weights of course. For this reason, when data is sequential and interdependent, such as weather forecasting, stock prices and most importantly language, normal neural networks don't perform well (Tealab 2018).

Recurrent neural networks (RNNs) solve this problem. They are in possession of an internal memory, which they use to process sequences of inputs, the sequences themselves hold more weight than the individual items. The internal memory allows them to take previous inputs into account to affect subsequent predictions. A corpus can be seen as one long sequence of data. This is why RNNs are especially useful when working on language-modelling tasks, when predicting a word, your model will be much more accurate if it considers what words came before it in a sentence.

The algorithm that is used for training neural networks is called backpropagation (Rumelhart, Hinton, and Williams 1985). Backpropagation calculates the derivative of the loss function w.r.t. each parameter in the network, it does this using the chain rule. In other words, the error is calculated between the predicted value and the actual value. To calculate how much each weight contributes to the overall error, the derivative will be calculated with respect to each variable in the equation.

In general it is advantageous to train neural networks with many layers, this will increase the learning capacity of the network and make it able to learn more complex datasets. The problem with neural networks with too many layers is that the gradient decreases exponentially as it's backpropagated through the network. This means that it's possible that when the backpropagation reaches the input, the gradient might be close to zero, and thus the weights will barely be updated (Hochreiter 1998). RNNs work much the same way, however the problem is amplified when working with sequential data: Previous elements of the input sequence are used as input for the next elements and the error is measured at each point in this sequence.

To give a better understanding of why this is happening; activation functions often 'squish' the output of a neuron into a space between 0 and 1, this is to introduce a degree of non-linearity into the network. A function that is often used for this is the sigmoid function, which is formally defined as

$$S(x) = \frac{1}{1+e^{-x}}$$

Its derivative is

$$S(x)' = S(X) - (1 - S(X))$$

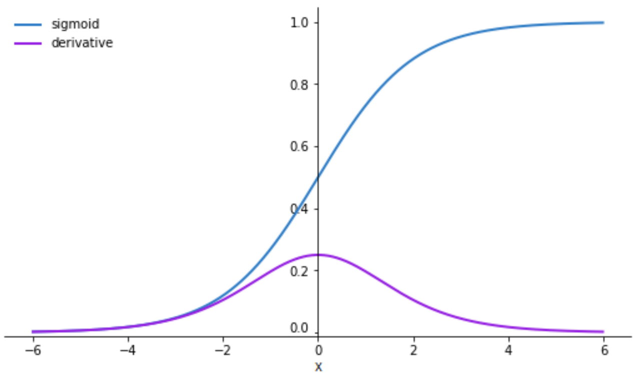


Figure 4.1: Sigmoid and its derivative

Values of x between -2 and 2 will have a much greater impact on the outcome than values outside of this range. This means that the gradient of that region will also be small (as can be seen in the derivative). If you have a long sequence of words, these gradients will vanish for words that came much earlier in the sequence. This means that these words will barely be taken into account, and so even for sequential data RNNs sometimes fall short.

A solution for this problem called 'Long Short Term Memory' (LSTM) was proposed by (Hochreiter and Schmidhuber 1997). An LSTM is a special version of an RNN, they have been proven to often outperform RNNs on many different tasks.

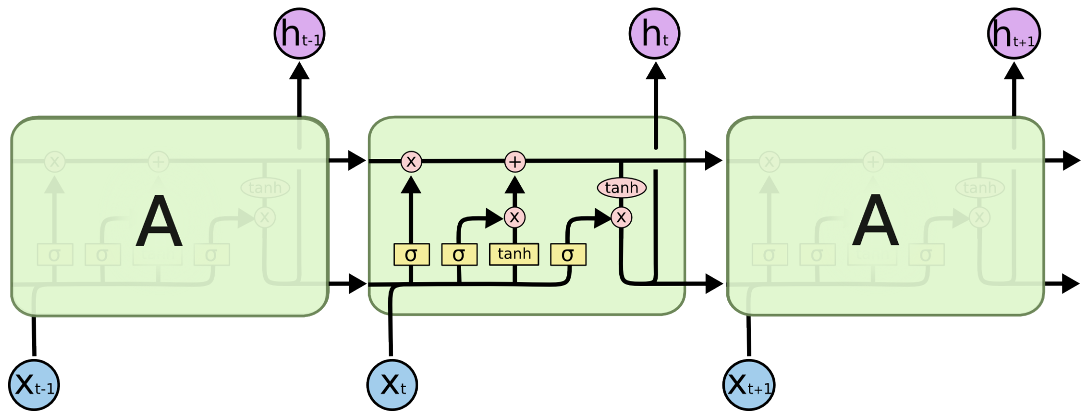


Figure 4.2: LSTM network

Figure 4.2 shows such a network. The line at the top of the diagram is called the cell state, the LSTM has the ability to add or remove information to this state, this is controlled by gates. The gates are the four yellow rectangles in the figure. The gates use two different activation functions: sigmoid and tanh. The tanh function is defined as

$$\tanh(x) = 2\sigma(2x) - 1$$

which, much like the sigmoid function, squishes the output into a space between -1 and 1.

The first gate on the left is the forget gate. It decides what information to keep from the cell state by using the current input vector (X_t) and the previous hidden state by feeding it into a neural sigmoid layer, every number in the cell vector is then multiplied by a number between 0 and 1. Where 0 means letting no information through and 1 letting all information through. For instance if a pronoun in the sequence is plural, but the new input is a singular pronoun, you want to forget the previous information about the plural pronoun.

After it has decided what information is going to be forgotten from the cell state. The tanh gate takes in the input X_t and previous hidden state, and creates new candidate values to be added to the cell state. The gate to the left of the tanh gate is called the input gate, which also takes in the input and previous hidden state, and outputs a vector whose values will be mutiplied by the candidate values of the tanh gate. This operation decides which candidates to keep, the result is then added to the cell state.

The rightmost gate is called the output gate, which is another sigmoid layer that outputs a vector that is then multiplied by the cell state that is put through another tanh function. The output gate decides which elements of the cell state to keep. For example, it might output information relevant to plurality, so if a verb comes next it knows what form it should have (Olah 2015).

LSTMs are able to capture long range dependencies that normal RNNs cannot. (Gulordava et al. 2018) utilised an LSTM in their research about long-distance agreement in various constructions. Their research showed that LSTMs were able to make accurate predictions of plurality over long sequences of data, not lagging far behind human performance. The researches hypothesised that LSTMs did not just extract patterns, but also acquired a deeper grammatical competence.

4.1 Parameter choices

4.1.1 Activation functions

A common activation function for the ouptut layer is the softmax function, which is defined as

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Where Z is the input vector, Z_i is an element of the input vector and K is the length (number of classes) of Z. Softmax is a more generalized version of the sigmoid function, it normalizes the output of the network to a probability distribution I.E. makes the final probabilities add up to 1. This is useful when making a prediction. This function is often used for multi class classification problems. The other activation functions that are used are the sigmoid and tanh functions.

4.1.2 Loss function

Categorical cross-entropy calculates the difference between two probability distributions, it is formally defined as

$$Loss = -\sum_{i=1}^{K} y_i log(x_i)$$

Where x_i is a probability returned by the model, and y_i is the corresponding target value. In practice y is all 0's, with a single 1 for the actual class. To minimize the loss, x_i would be 0 when y_i is 0, and 1 when y_i is 1. Categorical cross-entropy is well suited for classification problems and works well when using softmax as activation since it's comparing two probability distributions.

4.1.3 Optimizer

Adaptive moment estimation (Adam) (Kingma and Ba 2014) generally outperforms other optimization models (Ruder 2016). It computes adaptive learning rates for each parameter and uses averages to converge towards minima. The decaying averages of past gradients and past squared gradients are computed using the following formula

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where m_t is the mean of the first moment, and v_t the variance of the second moment. These are used to update the parameters

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon}$$

Where θ_1 are the parameters at timestep t, η is the learning rate and ϵ is usually a very small number to avoid divisions by zero. β_1, β_2 and ϵ were all kept to their recommended values of 0.9, 0.999 and 10^{-8} respectively.

4.2 Hyperparameter optimization

For every model different parameter combinations have been tried. Because of computational and time restraints this parameter search was rather shallow, however the combinations that were used in the end, have been empirically tested to be the most optimal ones for these corpora.

To reduce overfitting a regularization method called Dropout (Srivastava et al. 2014) was used, this involves approximating a large number of neural networks with different node architectures. During training there is a probability (1-p) of a node being temporarily removed, including it's connections. This makes it so each update to a layer is done with a different layer configuration. This introduces noise into the training, some nodes take on more or less responsibility. Complex co-adaptations, which usually don't generalize well to unseen data, can be broken this way.

Three different values of p were chosen: 0.9, 0.8 and 0.6. Other regularization methods were considered such as weight constraint and L1/L2 weight decay. However due their observed general ineffectiveness on this dataset and the constraints mentioned before, these were not used. Layer size was kept fixed at 256 for these same reasons, two layers were used for each model. Embedding layer size for each model was 200. Lastly the learning rate of the models had two different values: 0.001 and 0.002. This makes for a total of $3*2 = 6$ different combinations that were trained on each corpus. Adding any more dimensions to the search was not feasible since every extra dimension that is added grows the search space exponentially.

4.3 Vocabulary size

The vocabulary size of a model is simply all of the words (classes) it can predict. It is not feasible to simply equal this to the vocabulary size of the corpus (unique words in a corpus). The reason for this is that it limits model applicability because of computational and memory constraints (Yogatama et al. 2015; Faruqui et al. 2015). On the other extreme, making the vocabulary as small as possible, the measure of perplexity becomes trivial. Recall the perplexity equation from chapter 4. Say you have a vocabulary of 50, which will mean most of the vocabulary is made up of letters, with a completely uniform distribution (I.E. a model that has learned nothing). Every class will then have a probability of 0.02. Given enough data, the perplexity will converge to 50. This is the case for any number of classes. This means that with a smaller vocabulary size the perplexity will, in general, go down. This makes sense intuitively, there are simply less ways to be wrong, and the probabilities don't have to be spread out over as many different classes.

One might argue that the absolute value of the perplexity doesn't matter, only the relative differences between Zipfian and non-Zipfian corpora. However a reason not to do this is the increase in data size this would create, every word contains several letters, and consequently now occupies several data points. Another consequence of this is that long range dependencies are much harder to capture for the model, since the sequences become so much longer.

One advantage of using a character-level model is that you don't have the out-of-vocabulary problem, where some words not seen in the train data occur in the test data. This problem can also be solved by using subword encoding. The SentencePiece method was used for this purpose (Kudo and Richardson 2018). Subword encoding tokenizes parts of words of the corpus's vocabulary, if the model's vocabulary contains smaller subwords as well, previously unseen words can be stitched together from subwords and still classified. Another advantage of using this method is being able to create a link between words such as ’smart’ and ’smarter’, for example if the corpus contains a lot of adjectives, the algorithm is likely to pick up on subwords such as ’er’.

The question remains of what method to use to determine the vocabulary size of the model. Literature on choosing the right vocabulary size as it relates to corpora with different conformities to Zipf’s law is non-existent. Straightforward approaches could have influences on the outcome of the results due to the correlation between Zipf’s law, word frequency and vocabulary size of corpora. Before talking about how the decision was made, a few concepts require some explanation.

The vocabulary size of a corpus is highly correlated with the amount of words it contains. The growth of the vocabulary as related to the corpus size is described by Heap’s law (Heaps 1978), which is formally defined as

$$V_R(n) = Kn^\beta$$

where V_r is the vocabulary size for a corpus of size n. K and β are parameters that can be determined empirically. As n grows, the difference between n and $V_R(n)$ will grow with it. Meaning that as the corpus size increases, there will be more words that are repeated, and thus the growth of the corpus won’t contribute as much to the growth of the vocabulary. The general expected vocabulary size for different corpora sizes in Norwegian, which is the language of the corpora that have been used in this thesis, can be seen in figure 4.3. With parameters estimated using MLE (Vogelmann, Cornelissen, and Zuidema 2020).

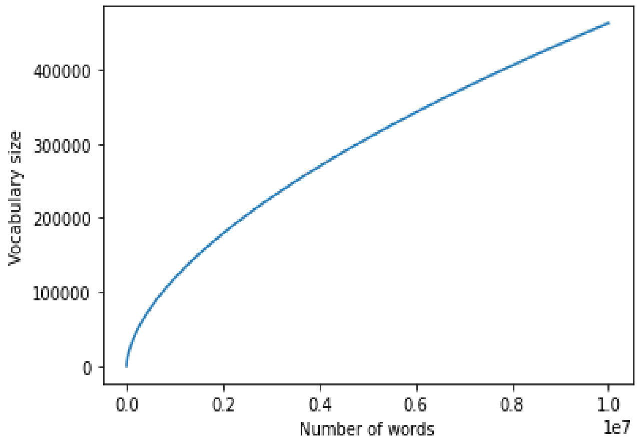


Figure 4.3: Heap’s law on the Norwegian Wikipedia corpus

Of course, this is what the plot looks like for a normal, Zipfian corpus. The vocabulary size of a corpus is also related to its Zipfianness. The typicality filter gives the text a more uniform word distribution, this means that low frequency words are assigned a higher probability, which translates to a higher vocabulary size. As you go further away from Zipf's law in a corpus, the distribution of word frequencies will get more uniform, the increased vocabulary growth that follows from this is thus an inevitable consequence of getting less Zipfian. This effect was seen in the corpora that were used in this thesis. Figure 4.4 shows the vocabulary sizes of the corpora used in this thesis, which contained 1 million words each.

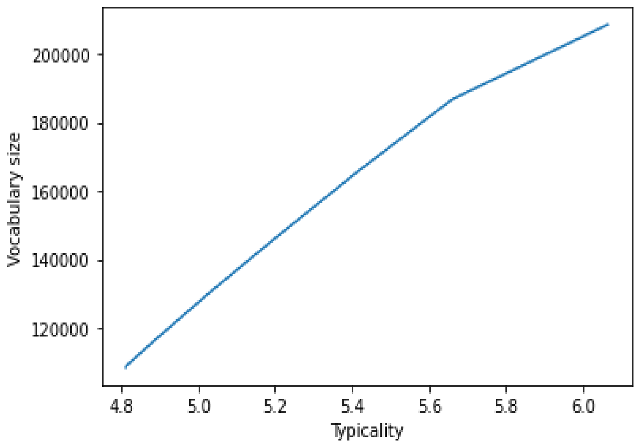


Figure 4.4: Vocab size by typicality

The vocabulary size seems to increase mostly linearly, although there is a slight deviation at the end of the curve. In the most extreme case, the non Zipfian corpora have a vocabulary about twice as large as the Zipfian corpora. When choosing the actual vocabulary size for the model, this rather significant difference has to be taken into account.

Several methods for choosing the vocabulary size were considered in this thesis, for all of which a case can be made. Making this decision wasn't easy as this is threading new ground, there is little to no source material on making this choice, so some assumptions had to be made. Especially as it relates to using SentencePiece, where the vocabulary choice has an extra dimension of difficulty, pertaining to the underlying algorithm. The vocabulary size you give is a parameter for the SentencePiece algorithm, this then has to be able construct each word in the corpus from that number. For different types of corpora, this results in different types of subwords, mainly their lengths and frequencies distributions will change. The effect that this has on the learning process has not been researched in this thesis, and could be a topic for future work.

- A frequency-based cutoff, this involves removing all words below a certain frequency (Luong, Pham, and C. D. Manning 2015). The vocabulary that the model would then use would consist of all the words that were not cut out. Given the high correlation word frequencies have with Zipf's law, this method was not used.
- Take a fraction of the corpus vocabulary size say, 5%, and use that as the model vocabulary size. As mentioned before a smaller vocabulary size usually coincides with a lower perplexity, this means that by doing this operation the comparison between a Zipfian and non-Zipfian corpus gets less meaningful.
- Keeping the vocabulary size at a fixed number.

Keep the vocabulary size fixed is what was chosen in the end. The main reason for this is that it doesn't have any of the aforementioned drawbacks. Nevertheless the problem remains of picking the exact number. This was done somewhat arbitrarily, from testing different sizes and evaluating the results, 6000 was chosen as a number that was not too big, not too small, and generally had a decent performance.

4.4 Text generation example

A model was trained on the (Zipfian, English) Brown corpus to give an indication of the performance and overall use of the models. The text was generated by giving the model a beginning of sentence (BOS) token, taking the highest probability word returned by the model, and use it as input. This is then repeated, continually generating text from its own outputs.

- *He was the only one who had had a good dealer without the car. He'd got plenty of time in a few days. I'm sure to be. I'm sorry to go out of the occupied space to be the same time. He had no idea that he would have to go out because she was not going. I saw the old men who were a good thing in the past. God's life. I'd better to do. He's a little time to keep a few minutes. I'd like to do. I was going to do anything about the time. Sarah said they are in the world. We're not to say and he was not going to do anything. He was the nephew. I had to go to it as a result of his own body. He's got out of his life and he'd got a little thing to go out. The oldest and the only thing to go to his feet to her. I've got a little sick. I had a chance to go to the door Sarah said. I'll see the way to the car. I'm going to do a good thing to go. Sarah said and he was not mad.*

The text, while not following a common thread, follows the basic grammatical structure of the English language, verbs, names and other nouns are all used properly. There are also very few non-existent words. The reason that non-existent words can be predicted at all is because of subword encoding.

5. Experimental setup

5.1 Subsampling

A language's true distribution is not known, there's only approximations, made by observing samples (corpora) of that language. A method used for making estimations like this is the Monte Carlo Method (Rubinstein and Kroese 2016). In the context of this thesis that means randomly sampling a set of corpora, and calculating the mean distributions. Doing this with enough samples, the estimated distribution approaches the actual distribution of the language. For this method you need a source of randomly sampled corpora, which are not easy to acquire.

A relatively new method called subsampling was developed for precisely this purpose (Vogelmann, Cornelissen, and Zuidema 2020). Subsampling solves the problem by using a single large corpus to serve the purpose of the sample source, each corpus is uniformly sampled from this large corpus. For instance the entire Wikipedia database in a certain language, which was used to sample the corpora in this thesis.

Zipf's law is an undeniable staple of human languages, this makes it nearly impossible to find natural corpora that have radically different typicalities. To acquire corpora with different levels of Zipfianness a typicality filter is applied while doing the subsampling. This filter keeps track of the typicality of the new corpus that is generated from the samples, for each sample (sentence) that is added, the resulting corpus is evaluated on its typicality. It can then be added to the corpus, or not, based on a tolerance parameter f (lower value means a higher tolerance) that decides when a sentence adds to the typicality of the corpus. The typicality is measured w.r.t. Zipf's law, whose parameters (α and β) have been estimated using a MLE of the original corpus C (ibid.). Figure 5.1 shows the influence of different typicality values on the rank-frequency relationship of corpora generated with the algorithm.

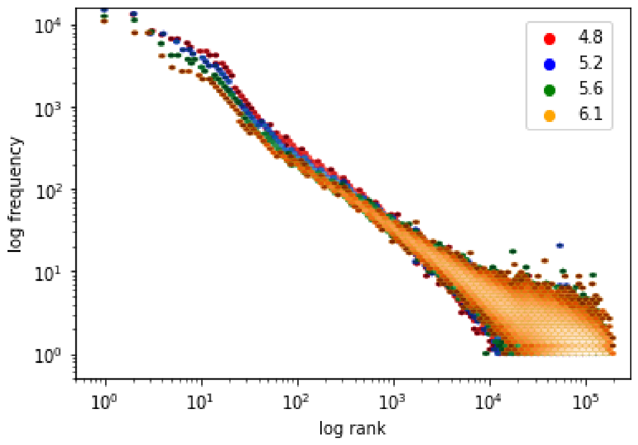


Figure 5.1: Effect on rank-frequency relationship of different typicality corpora

The lowest typicality of 4.8 is a uniform sampled corpus, meaning that it is like normal human language. As the typicality grows the word frequencies become more uniform, the rank-frequency relationship becomes flatter for higher typicality values. This means that the frequent words will occur less frequently, and the infrequent words will occur more frequently.

5.2 Corpora preprocessing

Using the Norwegian Wikipedia database as a large sampling source, 8 different levels of Zipfianness were used. The typicality ranged from 4.8 to 6.1, each with 10 corpora of around 1 million words, where each typicality value can be seen as its own 'language'. The corpora were cleaned of irrelevant characters and words (I.E. words from another language etc.). Choosing Norwegian as the language was mainly done because of availability, however Zipf's law is universal, it is present in every language. Therefore, even though some languages have different morphological complexities, which is related to the language's learnability (Blevins, Milin, and Ramscar 2017), the difference in learnability should hold.

Models such as LSTMs can't handle words as input, or anything other than numerical data for that matter. A common way of handling this in NLP is word vectorization, which is the process of encoding (sub)words as a number representation so that algorithms can use them for calculations. This is done by the SentencePiece algorithm.

5.3 Trainig models

Each of the different corpora have been individually trained until convergence using the python-based Keras implementation of an LSTM with TensorFlow as backend. A model was considered 'converged' if there was no increase in the validation loss for 5 epochs. This process was repeated 6 times, each for a different hyperparameter combination. The best performing algorithm was then chosen from these 6 different runs, and used as a surrogate of learnability. The code and data used in this thesis can be found here [1].

Each corpus contains subsampled sentences. For every sentence in the dataset a 'sliding window' was moved over it, this means that a sentence of 4 words will create 5 (or more, because of subword encoding) data points. The model will estimate the probability of the next element in a sequence given the context of the first part of the sequence.

$$P(w_i|w_{i-1}, ..., w_0)$$

Where i is ith word in a sentence. w_0 is the '0th' word in a sentence which marks the beginning of a new sentence. Consider the sentence:

- <S> They are playing soccer. </S>

Where <S> marks the beginning of a sentence and </S> the end of one. The sentence will then be split as follows:

Input data	Target to predict
<S>	They
<S> They	are
<S> They are	playing
<S> They are playing	soccer
<S> They are playing soccer	</S>

Sentences that were longer than 50 subwords were cut out of the corpus, this was done for two reasons: The first reason is a computational one, since sentences have varying lengths, shorter sentences all have to be zero-padded to be equal to the length of the longest data point. The average distribution of the sentence lengths can be seen in figure 5.2.

[1] https://github.com/11349875/NLP-lstm

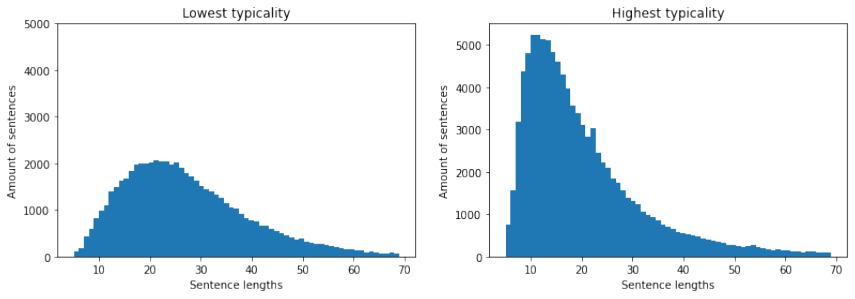


Figure 5.2: Average sentence lengths of corpora

Note the varying distributions for different typicality values, this is a consequence of the filtering algorithm that was used to create the corpora. Also note that the corpora with higher typicality had more sentences, which is a logical consequence of the sentences being shorter when the corpora are of the same size. As can be seen there are few outliers, on average only 2-4% of sentences are longer than 50 subwords, not much information is lost by this operation and the computational advantage alone was reason enough for this choice. The second reason is that after a certain input length, the first words of a sentence are barely considered by an LSTM, this is because of the vanishing gradient problem. Even though LSTMs were developed specifically to deal with this problem, it's not completely unaffected by it, given long enough input. Additionally sentences with lengths of 1 and 2 were cut out because these simply don't contain enough information to be relevant.

In the end the perplexity of a model was compared to the typicality of the corpus it was trained on, with this the effect of Zipf's law on learnability could be tested.

6. Results

Figure 6.1 shows the perplexity over the different typicality values. The blue line is the averaged values, every red x is one trained model's perplexity.

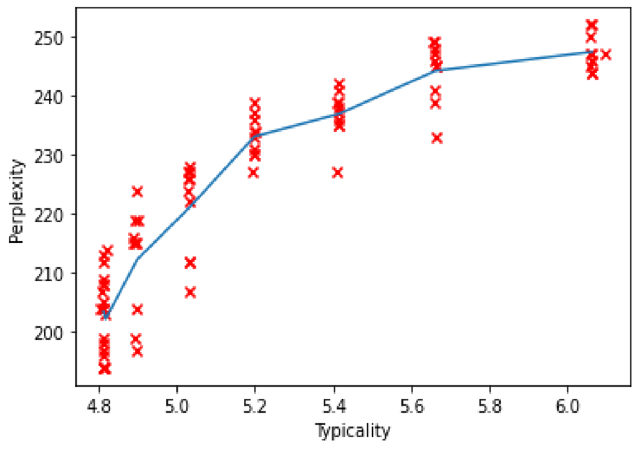


Figure 6.1: Perplexity vs typicality average

As can be seen in figure 6.1, there is a clear upward trend in the perplexity as the typicality gets higher, the results seem to increase somewhat logarithmically. The two distributions around a typicality value of 4.8 are both differently sampled, one uniformly and one with a very low f (the tolerance paremeter from the subsampling algorithm). The variance in perplexity of the lower typicality corpora is generally higher, this can be explained by the filtering algorithm that was used. When you subsample corpora without any restraints on which sentences you add to your corpora (I.E. sample them uniformly), there will be many different corpora that can be generated, these can, by chance, have differences in learnability that differ greatly from each other. When you are more selective with which sentences to add, there are far fewer possible corpora. This is also shown in practice, with a high typicality, corpora tend to overlap up to 27% on average (Vogelmann, Cornelissen, and Zuidema 2020). Because of this the decrease in variance is to be expected.

It is intuitive to think that a larger vocabulary would have a negative effect on the learnability of a language, after all there are simply more words to learn. When plotting the perplexity values against the vocabulary sizes it can be seen that the perplexity increases as the vocabulary size increases

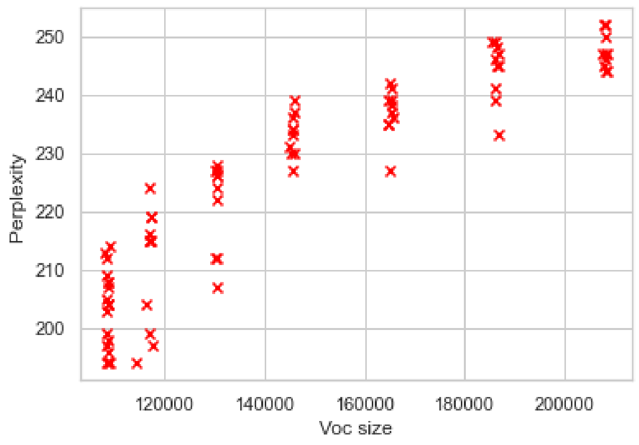


Figure 6.2: Perplexity vs vocab size

However since vocabulary size and typicality are highly correlated, it is hard to pinpoint the exact causes of the increase in perplexity. When looking at figure 4.4, it can be seen that the relationship between vocabulary size and typicality is almost linear. This means that the similarity of figure 6.1 and 6.2 are to be expected. Although vocabulary size is surely related to the perplexity, the amount of influence it has on it is not entirely clear.

The logarithmic increase of the line in figure 6.1 implies that after a certain point a higher typicality won't make for a higher perplexity. This is interesting because it shows that there might be more going on than just an increased vocabulary, if it was just the vocabulary size influencing the perplexity, you would expect to see a linear increase in perplexity as well, since typicality and vocabulary size have a linear relationship. Again, this is hard to confirm one way or another since it also might be the case that after a certain vocabulary size the perplexity isn't affected by it anymore.

6.1 Text generation

Looking at the differences between the trained models text generation gives us a different look at the performance of these models. Using the models best predictions it is possible to generate text. First the model is given only the beginning of a sentence: <S>, then it continually generates text using its own previous predictions. To avoid getting stuck in a loop, where its continually predicting the same sentences, one of the top four predictions was probabilistically chosen as the next word. Below is a snippet and its English translation, generated using a model that was trained on a uniformly sampled (Zipfian) corpus (with a typicality of 4.8).

- *Det var også et stort problem i europa. En og sørrheinfordrich ble det bygd et stort område. Verk og et stort område på grunnstater i den vestlige verden og i de fleste landeværninger ble drept i en analyse. Og ble dermed i 1863 utnevnte en ny del. Sembement. Og en av hans første album i den engelske klubben i den engelske kirkeprovins. S døde og var medlem i den tyske klubben i den norske kirke og ble senere kalte i 1774 og var i 1818 en professor. Og i det norske forbund. Slaget i den nye unionen i 1772 og i 1775. Og i 1859. Ssvåte og i en del. E som var blitt drept av en mannsavtale som var i drift. Og i 1799. Siske regiment. Og de fleste andre landskapsmedlemmen. Og andre landskapere ble opprettet i 1764 og det var et av den katolsk ortodoksste og politisk tid og det ble gjort en stor virkning av en mann som vartest i de første. E og i de neste tiåret i området som ifølge tradisjoner i et brev som ble styrt i et forsøk i en stor spekter for at det ikke er et av disse var de to år. Skeene som var en av dem. Og det ble gjort en stor figur for at det ikke er et av disse byense og de andre i verden er et av disse gruppens eldste plattform.*

- *It was also a big problem in Europe. A large area was built in Sørrheinfordrich. Works and a large area of land in the western world and in most land estates were killed in an analysis. And thus in 1863 a new part was appointed. Sement. And one of his first albums in the English club in the English church province. S died and was a member of the German club in the Norwegian church and was later called in 1774 and was in 1818 a professor. And in the Norwegian federation. The battle of the new union in 1772 and in 1775. And in 1859. Sweaty and in part. E who had been killed by a man contract that was in operation. And in 1799. Siske regiment. And most other landscape member. And other landscapes were created in 1764 and it was one of the Catholic orthodox and political times and it was made a big impact by a man who was last in the first. E and in the next decade in the area which according to traditions in a letter that was guided in an experiment in a large spectrum that it is not one of these were the two years. The spoons that were one of them. And a big figure was made that it is not one of these cities and the others in the world are one of these group's oldest platforms.*

Although the text is not very coherent, the general morphology of language is intact. Nouns are preceded by articles, adjectives precede nouns and verbs are used correctly to signal actions or events. There are few non-existent words and few repeated words. Below is a snippet of a model that was trained on a corpus with a typicality of 6.1.

- *Familien salaefamilien salicidae og zytaciaceae i gruppen nevinaceae i ordenenefamilien saledoniaceae og saturraceae i gruppen spinner. Idae i gruppen nephopidae i gruppen nephopidae i gruppen nephopidae er en artsrik fluere fageometrinestikke. Familien dvergere og naturgass i bregnefamilien innenfor sypresslekten i slekten caraca i ordenene. Slekten i blomstrer i norge og slektens hages.ere i usa og enefargetefamilien dvergseveks. Erfamilien isbreene. Slekten i havets. Erfamilien i slektens krypdyrfamilien i gruppens kultur. Erfamilien islopslekten og tilhører familien neovelopefamilien i slekten salecaticaceae i folkerepublikken8idaefamilien coleyiaceae og tilhører familiene fageometrisk spinner. Ophyllidae i idaefamilien columeyiaceae og sapiseaceae i russland og pakistanfamilien fabidae i ordenenefamilien salvinfamilien saleidae i slekten caravelae ipopeaceae og idaee idaee blomsterfluefamilien coleinaceae i eurthinaefamilien saledonidae i slektene coraceae i gruppen dvergspeiaceae i ordenen er en billeyger innenfor bladfamilien lyviniaceae i gruppen fugler i slektene skjoldfamilieni. Iaceae og svømmefamilien innenfor vepsene i slektene coridae istarslekten i gruppen nephophyllidae. Familien woodnoidae i storvinidae i familiegruppen løp.*

- *The family salaea family. salicidae and zytaciaceae in the group nevinaceae. in the order family saledoniaceae and saturraceae in the group spider. Idae in the group nephopidae in the group nephopidae in the group nephopidae is a species-rich fly fageometry stick. The family dwarfs and natural gas in the fern family within the cypress genus in the genus caraca in the orders. The genus i blooms in norway and the genus' gardens.ere in the usa and the monochromatic family dwarf. Erfamilien isbreene. The genus in the sea. The family in the family's reptile family in the group's culture. The family is the genus Spider and belongs to the family Neovelope family. In the genus Salecaticaceae in the People's Republic family Coleyiaceae and belongs to the families faGeometric spinner. Ophyllidae in idaefamilien columeyiaceae and sapiseaceae in Russia and Pakistan. Family fabidae in ordenenefamilien salvinfamilien saleidae genus caravelae ipopeaceae and idae flowers fly family coleinaceae in eurthinaefamilien saledonidae genera coraceae group dvergspeiaceae in order is a billeyger within blade family lyviniaceae group birds genera skjoldfamilieni. Iaceae and the swimming family within the wasps in the genus coridae istars genus in the group nephophyllidae. The family woodnoidae in storvinidae in the family group ran.*

The text generated by this model can be intuitively seen as less coherent. Although the grammatical structure is still present, the amount of nonsense words in between makes it hard to read even the individual sentences. There are many repeat words, that seem to appear randomly. In this particular text you can find the word 'family' in almost every sentence. This pattern was fairly consistent for models that were trained on the corpora with a typicality of 5.5 or higher, although the words themselves were different. What's more, even with the probabilistic way of choosing words, the model often falls into a repeat where it's continually generating the same sentences. Additionally, the high typicality models are much more likely to predict an end of sentence (EOS) token, creating many 1 and 2 word sentences, which is not as visible in this snippet but present in most generated texts. Lastly, the high typicality models were a lot more sensitive to noise, not all words of a different language could be easily removed from the data, sometimes the models will predict a string of German words for instance, this does not happen for the lower typicality models.

The models were also used to generate a small corpus of 100.000 subwords for the Zipfian and non-Zipfian models. The thought behind this was to see how the texts distributions would differ from the corpora they were originally based on, which is an alternative to perplexity to measure a models performance (Meister and Cotterell 2021). Each of the models that generated the texts have their own 'language' (each typicality value could be seen as its own language) meaning that they would have to be compared to distributions of language they were trained on. Otherwise you would be comparing a model trained on a non-Zipfian language, to a Zipfian language, which would defeat the purpose.

The first property that was tested was the conformity to Heap's law. Both of the differently trained models have a vocabulary size much lower than what is to be expected according to Heap's law. Despite the repeating words and sentences of the high typicality models, their vocabulary size was much higher than the ones of lower typicality models, meaning that they more closely resembled Heap's law, and thus learned the language better. However this is misleading, the high typicality model cheats by creating non-existent words, this makes the comparison between the two null.

This problem combined with the repeating sentences which can make up half of the corpus and the frequent predictions of EOS tokens, makes comparisons to other types of distributions dubious at best. The high typicality text generation is simply so bad that any comparison to it becomes borderline meaningless. The only way to be able to make a comparison between the two models would be to add a lot of caveats, such as making the generation even more random, or only letting the model predict an EOS token if it hasn't done so for a while. However the comparison won't be objective when doing this, since all of the appendages will be geared towards helping the higher typicality models.

6.2 Kolmogorov complexity

Apart from measuring the perplexity of the trained language models and looking at text generation, there are other ways to estimate the learnability of a language. One other way that has been explored in this thesis is by approximating the Kolmogorov complexity of corpora sampled from that language. The Kolmogorov complexity of x is defined as (M. Li, Vitányi, et al. 2008)

$$K(x) = \text{the shortest computer program (in binary) that generates x}$$

The right hand side of this equation can be seen as the shortest description or ideal compression of x. To give an understanding of the different complexities that x can have, consider the following strings:

- 000000000000000000
- 000000000111111111
- 001011000110111011

Intuitively, the first string looks less complex than the second one, and the second one looks less complex than the third one. The first one could be easily described as " 18 * '0' ", the second one as " 9*'0'+9*'1' ". For the third string finding a shorter way to describe it is a lot harder, this string is intuitively more complex than the other two.

The actual shortest way to describe these strings cannot be known however. This gives way to an important property of Kolmogorov complexity: it is not a computable function. This is because of the halting problem (Turing 1937). It can however, be approximated. Standard compression programs such as 7-Zip and WinRAR serve this purpose (M. Li, Chen, et al. 2004).

A simplicity based learner is a learner who's only learning bias is simplicity. Some previous research points in the direction that humans are simplicity based learners. Given sequence of words this learner predicts the continuation based on what will minimize the Kolmogorov complexity, the performance of the learner is then measured in terms of the prediction error. This error will be lower when the Kolmogorov complexity of the language which produced the sequence, is lower. The learnability of that language is then inversely related to its Kolmogorov complexity (M. Li, Vitányi, et al. 2008).

The same corpora used to train the models were examined for their approximated Kolmogorov complexities. The original file sizes for each corpus varied, so the size of the compressed files were normalized against the original size of the corpora. Two different compression algorithms were tested: the first one is ZIP-FILE, which uses the 7zip algorithm, which compresses the file without any loss of data. The second algorithm that was used was LZMA (Lempel–Ziv–Markov chain algorithm), (Ziv and Lempel 1977) which also compresses the file without loss of data. Note that any algorithm that approximates Kolmogorov complexity always overestimates it (underestimating it would be impossible), and thus underestimates learnability.

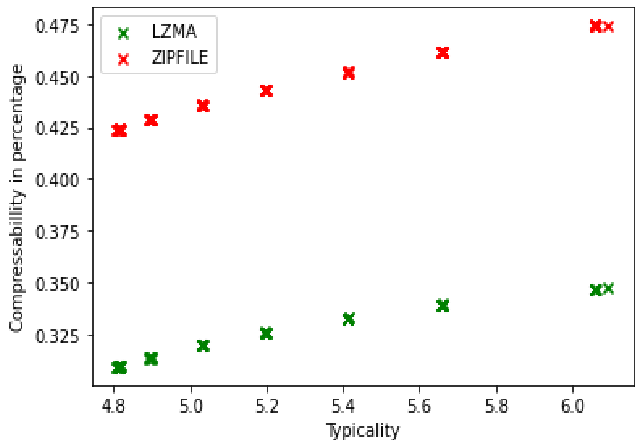


Figure 6.3: Compression size vs typicality

On the y-axis is the percentage of the original file size the compressed files are. Note that some of the x's are several data points, they look like one data point because there is very little variance. As figure 6.3 shows, as the typicality of a corpus grows, so does the compressed size of that corpus. The estimated Kolmogorov complexities of corpora with a higher typicality is therefore higher. The LZMA algorithm outperformed the ZIP algorithm, their relative differences are almost identical however. Figure 6.4 shows the file sizes, normalized by the smallest value of the respective algorithm.

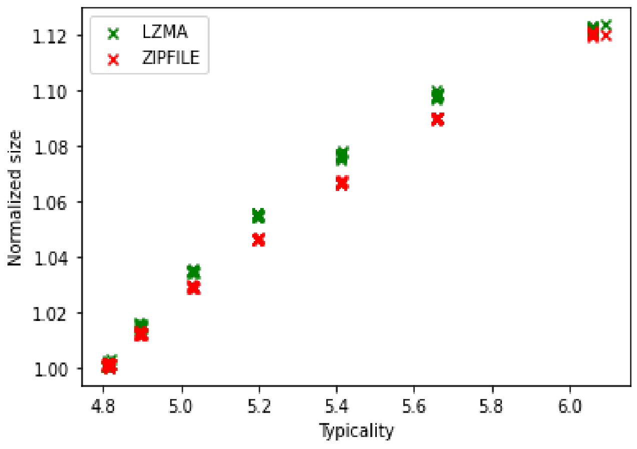


Figure 6.4: Normalized compression size vs typicality

This shows that the result is overall robust, different approximation methods yielded the same outcome. From these graphs it can be concluded that as the Zipfianness of a language decreases, the estimated Kolmogorov complexity increases. Since complexity is the inverse of learnability, this strengthens the previous results of the LSTMs. As a language gets less Zipfian, the estimated complexity goes up, and consequently the learnability goes down. The increase in vocabulary size and morphology are likely to blame for this increase in complexity. Interesting to note is that although the variance of perplexities was high in the lower typicality ranges, the variance of the compressed file size is very low across all typicality values.

Another reason for the decrease in compressibility could be that the average word length of the Zipfian corpora was 5, compared to the 6.1 of the non-Zipfian corpora. Longer words carry more information (Newman 2005). This means that on average the words in the non-Zipfian corpora are more complex, and harder to learn.

Some research suggests that having shorter more frequently used words, helps with spreading out information load density for listeners. Common low information words, spread out the higher information words, so that the information flow stays constant (Piantadosi, Tily, and Gibson 2011). With non-Zipfian languages, this becomes harder to do as there are more high information words, and less short words to tie them together. Finding the relevant higher information words in a sentence will be more difficult because of this.

7. Conclusion

The results overall clearly show that Zipf's law affects learnability, furthermore it shows that it has a positive effect on the learnability of language. When Zipf's law is less present in a language, the learnability decreases. This result is confirmed by the increase in perplexity, the decrease in compressibility and the differences in text generation, although this is less concrete.

As touched upon in the results, the vocabulary size is likely to be one of the factors of the increase in perplexity, maybe even the largest factor. However even if vocabulary size rather trivially is the only reason for the increased perplexity this is still a result of the differences in Zipfianness, and therefore Zipf's law would still have a positive effect on learnability.

Another reason besides this could be because of the increase in morphology that the non-Zipfian languages have, there are a lot more syllables and combinations of those that have to be learned, this also shows in the text generation. The text generation has a lot of trouble with determining what even constitutes a word, and often misplaces the wrong syllables together. This finding points in the direction of the previous research done by (Kurumada, Meylan, and Frank 2011) who hypothesised that Zipf's law helps with word segmentation. The richer morphology of non-Zipfian languages could also contribute to the difficulty the models have with determining when the end of a sentence is likely to come.

The difference in generation of text might be a consequence of the higher perplexity of the high typicality models, it doesn't necessarily mean that it's a direct consequence of the different levels of Zipfianness. However, the difference in perplexities are a consequence Zipfian levels, so it still establishes a link between learnability and Zipfianness. Furthermore the difficulty of creating a valid corpus with the high typicality models are also indicative of the decrease in learnability, it seems that with a lowered presence of Zipf's law, the language becomes too complex and vast to learn.

The Kolmogorov complexity of the corpora further confirm the results that non Zipfian-languages are harder to learn. The lower (estimated) complexity of the more Zipfian corpora indicate that these languages are easier to learn, as complexity is the inverse of learnability. The increased richness of the morphology is likely to be one of the main culprits of the increased complexities of the non-Zipfian languages, it is harder to establish a pattern in the data when the pattern is more complicated. The different word lengths that arise from Zipf's law could also be a culprit, longer words contain more information and are more complex, which makes them harder to learn. Additionally, shorter words help with spacing out longer words, making them easier to learn.

On top of the results on learnability, this thesis lends credibility to George Zipf's theory on the origin of Zipf's law, where he hypothesised that the clash between speaker and listener is the cause. The more uniform the word distribution is over a language, the less learnable and more complex it becomes (which is what the listener would want, to understand what was being said), the steeper the distribution becomes over a language, the more learnable it becomes (which is what the speaker would want, to convey his thoughts in as few words as possible). However if the language is too steep, it loses its communicative value because too many words will be low information, filler words. And as shown in this thesis, if the distribution is too uniform, the language loses its learnability. This is why Zipf's law might be the perfect distribution over languages, where it's not too hard to learn, but still has communicative value.

In conclusion, Zipf's law doesn't just influence learnability, it influences almost every aspect of language: sentence lengths, word frequencies and their distributions, word lengths and vocabulary size. It is not surprising then, that learnability is also be affected by it.

In this thesis different ways of estimating the learnability of languages have been tested. All of the different measures pointed in the same direction: That Zipf's law has a positive effect on learnability. Although the exact reason for this is hard to pinpoint, it likely has to do with the increased morphology of non-Zipfian languages and the increased vocabulary size. Both of which make a language more complex, and harder to learn.

8. Discussion and Future work

This thesis was done on Norwegian corpora, the easiest and most relevant research is to apply the work done in this thesis to more languages, which differ in their morphologies and complexities. In addition the models could be trained with a larger parameter search, layer size, embedding layer etc. Every part of this thesis that was limited by computational constraints could be further explored without these constraints.

As mentioned before, there was some difficulty relating to choosing the vocabulary size. This choice was difficult mainly because there was no prior literature on the choosing a vocabulary size for corpora with different conformities to Zipf's law. Furthermore, when using subword encoding combined with these different conformities, which affects sentence lengths and word lengths of the encoded data, this choice is not trivial. It's possible that these differences affect performance significantly when tested properly, or it might be that they differences are marginal, either way more research is needed on this topic.

Perhaps when text generation for the high typicality models is better the distributional properties of the generated texts can be compared to each other. Finally, the root causes for the effect of Zipf's law on learnability have been explored in this thesis, but not thoroughly tested or analyzed. Some analyses could be done on the significance of different variables using methods such as linear regression. It would be interesting to see these causes be further researched in the future.

9. Acknowledgements

I would like to thank Jelle Zuidema for his consistent interest and guidance. I'd like to also thank the projects collaborators Bas Cornelissen and Oskar van der Wal for their time. Lastly I want to thank Valentin Vogelmann for being so invested in and willing to help with the project, you've been a great help these past few months and I thoroughly enjoyed discussing the concepts and ideas behind the project with you. Without you the project would not only have been a lot harder, but also less enjoyable.

Bibliography

Berndt, Bruce C (1972). “On the Hurwitz zeta-function”. In: *The Rocky Mountain Journal of Mathematics* 2.1, pp. 151–157.

Blevins, James P, Petar Milin, and Michael Ramscar (2017). “The Zipfian paradigm cell filling problem”. In: *Perspectives on morphological organization*. Brill, pp. 139–158.

Blythe, Richard A, Kenny Smith, and Andrew DM Smith (2010). “Learning times for large lexicons through cross-situational learning”. In: *Cognitive Science* 34.4, pp. 620–642.

Cancho, Ramon Ferrer i and Ricard V. Solé (2003). “Least effort and the origins of scaling in human language”. In: *Proceedings of the National Academy of Sciences* 100.3, pp. 788–791.

Chawla, Nitesh V (2009). “Data mining for imbalanced datasets: An overview”. In: *Data mining and knowledge discovery handbook*, pp. 875–886.

Christiansen, Morten H and Nick Chater (2008). “Language as shaped by the brain”. In: *Behav Brain Sci* 31.5, pp. 489–509.

Cover, Thomas M (1999). *Elements of information theory*. John Wiley & Sons.

Faruqui, Manaal et al. (2015). “Sparse overcomplete word vector representations”. In: *arXiv preprint arXiv:1506.02004*.

Gulordava, Kristina et al. (2018). “Colorless green recurrent networks dream hierarchically”. In: *arXiv preprint arXiv:1803.11138*.

Heaps, Harold Stanley (1978). *Information retrieval, computational and theoretical aspects*. Academic Press.

Hendrickson, Andrew T and Amy Perfors (2019). “Cross-situational learning in a Zipfian environment”. In: *Cognition* 189, pp. 11–22.

Hochreiter, Sepp (1998). “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02, pp. 107–116.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Johnson, Justin M and Taghi M Khoshgoftaar (2019). “Survey on deep learning with class imbalance”. In: *Journal of Big Data* 6.27, pp. 1–54.

Jurafsky, Dan and Christopher Manning (2012). “Natural language processing”. In: *Instructor* 212.998, p. 3482.

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kudo, Taku and John Richardson (2018). “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing”. In: *arXiv preprint arXiv:1808.06226*.

Kullback, Solomon and Richard A Leibler (1951). “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1, pp. 79–86.

Kurumada, Chigusa, Stephan C Meylan, and Michael C Frank (2011). “Zipfian word frequencies support statistical word segmentation”. In: *Proceedings of the Annual Meeting of the Cognitive Science Society* 33.33.

Li, Ming, Xin Chen, et al. (2004). “The similarity metric”. In: *IEEE transactions on Information Theory* 50.12, pp. 3250–3264.

Li, Ming, Paul Vitányi, et al. (2008). *An introduction to Kolmogorov complexity and its applications*. Vol. 3. Springer.

Li, Wentian (1992). “Random texts exhibit Zipf’s-law-like word frequency distribution”. In: *IEEE Transactions on information theory* 38.6, pp. 1842–1845.

Luong, Minh-Thang, Hieu Pham, and Christopher D Manning (2015). “Effective approaches to attention-based neural machine translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421.

Mandelbrot, Benoit (1953). “An informational theory of the statistical structure of language”. In: *Communication theory* 84, pp. 486–502.

Meister, Clara and Ryan Cotterell (2021). “Language Model Evaluation Beyond Perplexity”. In: *arXiv preprint arXiv:2106.00085*.

Newman, Mark EJ (2005). “Power laws, Pareto distributions and Zipf’s law”. In: *Contemporary physics* 46.5, pp. 323–351.

Olah, Christopher (2015). *Understanding LSTM Networks*. URL: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. (accessed: 24.07.2021).

Piantadosi, Steven T (2014). “Zipf’s word frequency law in natural language: A critical review and future directions”. In: *Psychonomic bulletin & review* 21.5, pp. 1112–1130.

Piantadosi, Steven T, Harry Tily, and Edward Gibson (2011). “Word lengths are optimized for efficient communication”. In: *Proceedings of the National Academy of Sciences* 108.9, pp. 3526–3529.

Rubinstein, Reuven Y and Dirk P Kroese (2016). *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons.

Ruder, Sebastian (2016). “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Tealab, Ahmed (2018). “Time series forecasting using artificial neural networks methodologies: A systematic review”. In: *Future Computing and Informatics Journal* 3.2, pp. 334–340.

Turing, Alan Mathison (1937). “On computable numbers, with an application to the Entscheidungsproblem”. In: *Proceedings of the London mathematical society* 2.1, pp. 230–265.

Vogelmann, V., B. Cornelissen, and W. Zuidema (2020). *Statistical Methodology for Quantitative Linguistics: A Case Study of Learnability and Zipf's Law.*

Yogatama, Dani et al. (2015). “Learning word representations with hierarchical sparse coding”. In: *International Conference on Machine Learning*. PMLR, pp. 87–96.

Zipf, George Kingsley (1949). *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books.

Ziv, Jacob and Abraham Lempel (1977). “A universal algorithm for sequential data compression”. In: *IEEE Transactions on information theory* 23.3, pp. 337–343.