which means "surround with a rampart or other fortification" **circumvallate**

Hello, my name is Clemens Binz, and my Bachelor thesis topic is “Exploiting Hierarchical Structure in Time Series Data.”

Since 2009 Google has continuously scanned Books and published this data on the Google Books Ngram Corpus. The Dataset is based on Books from around the 1500s up to 2019 and consists of Ngrams that appear at least 40 times across the corpus. The corpus consists of n-grams beginning with 1-grams, also known as a unigram and extend up to 5-grams. The Data is freely available as plain text in comma separated Value format like in the following example: first the n-gram Word combination followed by the Year, in this case 1978, the 335 occurrences of this n-gram in that year overall and lastly the 91 distinct books in which the n-gram was found.

The resulting time series based on this data can be of use for historians or for linguistic purposes. This Plot of the Relative frequency of the 1-gram Revolution in relation to famous historic events may be a concrete use case for the Google Books Ngram Corpus Data. The rise in occurrences of the word revolution can help in understanding why a revolution has happened, what the origin or trigger of the revolution was or what indicates a possible future revolution.

If we take a closer look into the Dataset as a whole, then it becomes clear that larger n-grams dominate the space requirements for every language that is a part of the Corpus provided by Google. Englisch works were focused but German and French 5-grams have also reached Terabytes of storage needs.

The Corpus has unique properties such as space requirements growing in relation to n and data from natural language. Therefore, with existing dependencies between different n-gram time series based on grammatical rules which can be exploited by our Approach for time series Compression.

Time Series compression is a well-researched topic, and the most popular approaches can be categorized into 4 Sections as well as Lossy or Lossless compression. Piecewise or Dictionary-based approximation is based on the principle that time series share some common segments. These segments can be extracted into atoms, such that a time series segment can be represented with a sequence of these atoms. Some state-of-the-art techniques in this Section are CORAD or SAX. The main idea behind function approximation is that a time series can be represented by a simpler function similar to interpolation which needs less space. Here Discrete Fourier Transform, Discrete Wavelet Transform and Piecewise Polynomial Approximation are prime examples. An autoencoder like RNNA or LFZip are particular neural networks that are used to compress data by learning a representation of the input data and then attempting to reconstruct the original data from this representation. And lastly Sequential Algorithms combine different compression techniques after each other to achieve higher compression. Popular choices include Huffman, Delta or Run-length encoding in combination with some kind of novel compression method.

In contrast to these techniques, we intend to use our existing domain knowledge as leverage for a new compression Approach.

We established that our data is based on natural Language and has inherent dependencies between similar n-gram time series. If we take a look at the two functions for the 3gram "United States of” and the 4 Gram “United States of America”, then these dependencies become clearly visible to the naked eye. There probably exists some kind of scaling to only store one of these functions for this and other similar examples in the Google Books Ngram Corpus.

The first Observation we made analyzing the corpus is that the first (or last) n − 1 tokens of a n-gram severely restrict the set of possible last (or first) tokens. This information gain increases with higher n-grams because correct and sensible sentence structure restricts possible new tokens.

This is where our Hierarchical Compression Algorithm was formed. And it goes as follows: for a given set of n-gram frequency series T = {(g0, τ0), . . . ,(gn, τn)} and a given n-gram gi , find gl (gr ) by removing the last (first) token from gi , and minimize ε = λ(τi , αl · τl + αr · τr) for a given loss function λ. If ε < δ for a given error bound δ, store only αl and αr instead of τi. Therefore, n-gram times series of higher order can be stored as linear combinations of n-1 gram time series.

Here u can see our previous 4gram time series “United states of America” next to an example scaling of the two 3grams “United States of” with scaling factor 0.25 and “Sates of America” with a factor of 0.75. Which merge into nearly identical functions after 1975.

In order to bring this approach to live some Data Preprocessing steps were necessary. The Corpus has some characteristics that need to be accounted for. If we take a look into the distribution of the total counts of all unigrams in relation to the available time axis, illustrated by this graphic, then one becomes visible. Data is very sparse in the early centuries and most data gathers in the second half of the available time span. If we followed our approach for approximating time series data on the full time scale then the vast amount of 0 entries for early years were no occurrences of an n-gram were documented will impact the quality of the approximation for later years were occurrences accumulate. This resulted in our decision to Sample the data and focus on years after 1800. We also choose to limit the data up to the year 2000 because google choose to integrate e-books into the corpus which resulted in old books being reintroduced into the dataset as e-books screwing with integrity of the data as well as an overrepresentation of publishers and generally less moderated data. The last step needed was to reintroduce the missing zero entries which are implicit and therefore not stored in the raw data. After all these Steps the first time series could be approximated.

This is the Zscore Normalized approximation of the German 2-gram “Petri stand” following our concrete implementation which will be presented in the following slides. However, in order to explain the reason why the approximation is normalized we first need to introduce some metrics on which our approach was constructed as well as the following experiments.

Experiments are based on the two metrics: Compression rate and Root mean square Error as distance function between our approximated function and the original. These were chosen as the preferred metrics in many lossy states of the art time series compression techniques and therefore also fit for our purpose. We compared Compression Rates to different allowed error rates previously set as hyper parameter. Root mean square error is a metric that heavily punishes larger error but is not suited for comparison of low and high frequency data which does include our Google Books Corpus. Overall occurrences of Ngrams can range from 40 up to 10^9. To solve this problem, we chose to normalize our data. Zscore normalization was a technique already used by some of the techniques we were evaluating against and was therefore deemed as a good fit. Each frequency time series was scaled by the mean and standard deviation of the series achieving a Zscore normalization with a mean of 0 and a standard deviation of 1.

Now that all metrics are introduced, we can look at our concrete implementation of the previously explained hierarchical compression approach.

We used pyspark in combination with the parquet file format in order to handle these large data files through distributed in memory processing. First, we load the n-gram data filter out any unigram which can not be compressed by the approach. Then we search for the two children of each n-gram and calculate the approximation, coefficient and the intercept through Multilinear Regression. Next, the approximation needs to be Zscore normalized to calculate a meaningful and comparable Root mean square error. And finally, if the error is below the threshold the time series can be replaced by the 2 child coefficients an intercept and two references to the child time series as well as an identification for the original n-gram. If this process is followed the n-gram frequency time series can be represented by 6 variables under a given error constraint.

And the reconstruction process is quite similar. The time series can be built through references and coefficient’s with the same formular.

One observation that has to be noted is that this approach introduces stacking errors which affect the overall effectiveness of the compression. 5 gram approximations for example are built from 4 grams which might be approximations build from 3 grams and so on with each layer introducing a new error which builds upon the previous one.