Mutual Information Using LZW-Compression

Noah S. Peterson
Dept. of Mathematics & Computer Science,
Drake University, Des Moines, IA, 50311, USA

The Lempel-Ziv-Welch Compression Algorithm is an extremely fast and information lossless method of compressing data. Mutual information is the information shared between two objects. In this paper we proposed several possible definitions of Mutual Information using LZ compression in hopes that we can find a fast and efficient algorithm for Computing mutual information. We proceeded to run these definitions over a range of sequences to test their viability definitions for mutual information. We then applied these definitions to Protein Sequence Alignment as a possible practical application. Finally we produced an allegorically generated sequence that is a worst case scenario for Lempel-Ziv-Welch Compression that we believe will help us improve the definitions we proposed.

Contents

I. BACKGROUND

A. Lempel-Ziv-Welch Compression Algorithm

The Lempel-Ziv-Welch Compression Algorithm (LZ) is an information lossless method of data compression created by Abraham Lempel, Jacob Ziv, and Terry Welch. It is currently used in Unix based compression utilities and GIF image files [1].

Other methods of encoding often create a dictionary of encoding which the decoding device must use to reassemble the original object. What makes LZ so efficient is that it creates its compression dictionary iteratively in such a way that it can be decoded iteratively with out actually sending the compression dictionary to the decoder [2] [3] [4].

For example let's take the simple sequence "0101". In Table I we can see how this is encoded.

Current Value	Next Character	Encoding	Add To Dictionary
0	1	0	01 as 10
1	0	01	10 as 11
0	0	10	

TABLE I: Shows step by step encoding of sequence "0101"

This process encodes "0101" as "00110". This is not a very efficient but it displays a few things which are important strengths and downsides of the algorithm. First note that strings which have been seen before are kept and stored for later use meaning that as sequences are repeated the compression gets more efficient. This allows the algorithm to take advantage of reoccurring sequences which often occur in the data we want to store. Also note that when we encoded the digit 1 we encoded it as 01, this is because in order to be able to decode these values they all must be of a standard length and in order to encode for 3 possible values we need sequences of size at least 2 in binary. For a little more efficient encoding we can look at the example in Table II.

Current Value	Next Character	Encoding	Add To Dictionary
0	0	0	00 as 10
00	0	10	000 as 11
000	0	11	

TABLE II: Shows step by step encoding of sequence "000000"

This process encodes "000000" as "01011". Still not very efficient but we can see how LZ takes advantage of redundancy, in longer sequences it typically works much better.

The version of LZ we use for this project has a few differences to give it a little more consistent encoding and is helpful in our proposed definitions. First, it has no limit to its maximum dictionary size. It also uses a look ahead value when adding values to the dictionary which generally makes it a little more powerful compressor.

B. Compression Ratio

The compression ratio we define as the length of the compressed sequence over the length of the uncompressed sequence $\rho_{LZ}(s) = \frac{|LZ(s)|}{|s|\log_2(|a_i|)}$. For example the compression ratio of the process from Table I would be $\rho_{LZ}(\text{"0101"}) = \frac{|LZ(\text{"0101"})|}{|0101|} = \frac{5}{4}$ ($LZ(\text{"0101"})$ is the sequence encoded by LZ with "0101" as input) whereas the process in Table II would be $\rho_{LZ}(\text{"000000"}) = \frac{|LZ(\text{"000000"})|}{|000000|} = \frac{5}{6}$

C. Mutual Information

Mutual information is the information shared between two objects. Information theory often describes the information in an object as its entropy (H). There are a few ways we can define the shared or Musial Information. First is $H(X) + H(Y) - H(X, Y)$. This is to say we take the information content of X plus the information content of Y and subtract the joint Information content. The joint information content is information contained in the union of

X and Y if you'll pardon the set theory analogy. Thus as we can see from Figure 1 if we add $H(X) + H(Y) - H(X, Y)$ should give us only the overlapping region of H(X) and H(Y) [5][6]. We can also compute it using what is known as the conditional information H(X—Y) i.e. all the information in X that is not also in Y. In Figure 1 this would be the crescent of H(X) that does not include I(X:Y). Thus we can also define mutual information as $H(X) - H(X|Y)$ or $H(Y) - H(Y|X)$ [5][6]. We also generally relate the information

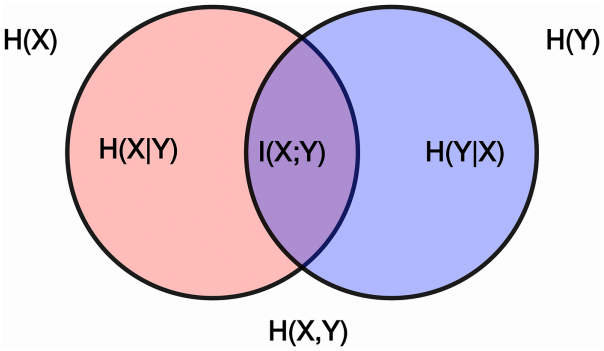

FIG. 1: Visual representation of the mutual information of two objects [7]

content of an object or sequence to the to its length when compressed. The idea being that when an object is compressed we have taken out all of the redundant information so we can see its true information content. Thus we have the following general definitions of information theory using the compression ratio [8].

$$I(X : Y) = H(X) + H(Y) - H(X, Y) \tag{I.1}$$

$$I(X : Y) = H(X) - H(X|Y) \tag{I.2}$$

$$I(X : Y) = \rho(X) + \rho(Y) - \rho(X, Y) \tag{I.3}$$

$$I(X : Y) = \rho(X) - \rho(X|Y) \tag{I.4}$$

II. MUTUAL INFORMATION USING LZ

Our project was to see if we can use the LZ compression ratio to compute or approximate the mutual information of two objects. To do this we tried basing some definitions for LZ Mutual information based off the definitions from equations I.3 and I.4.

A. Our Definitions of LZ Mutual Information

Unfortunately defining the Joint and conditional compression is not easy when you are actually trying to apply an algorithm to sequences. We tried a few different methods of finding the joint information using LZ and then used those definitions to emulate the definition in Equation I.3. Our first two approaches were fairly simple, concatenating X and Y together, and took the LZ compression Ratio of that (See Definition 1). For our second approach we "zipped" X and Y together alternating bits and ran that through the LZ compression ratio (See Definition 2).

Our final definition of joint LZ compression is a bit more intricate. As stated in Section I A LZ creates a dictionary which contains substrings of the sequence it is compressing and by the end of the compression this dictionary contains a set of substrings which could be used to generate the initial sequence. In essence this dictionary contains all the information of the sequence. So we defined the algorithm $\rho_{LZCrossed}(X, Y)$ as fallows: First compress X and Y separately using LZ. Instead of taking the encoding from these operations we will take the dictionaries created by these presses. Then we once again compress X and Y using LZ except for this time we give LZ the dictionary generated by Y when it is compressing X and vice versa. Thus we generate a compression of each sequence based on the information of the other sequence. Finally we take these two encodings, concatenate them together and return the ratio of their length to the length of the staring sequences. We use this algorithm for the joint information in Definition 3; however, we also noted that it may have some relevance to con-

ditional information rather than joint information. Thus for our final definition (4) we based on Equation I.4 using $\rho_{LZCrossed}(X|Y)$ as the conditional compression. Note that $\rho_{LZCrossed}(X|Y)$ is similar to $\rho_{LZCrossed}(X, Y)$ except we only use the dictionary of X to compress Y.

$$I(X : Y) = \rho_{LZ}(X) + \rho_{LZ}(Y) - \rho_{LZ}(X + Y) \quad (1)$$

$$I(X : Y) = \rho_{LZ}(X)+\rho_{LZ}(Y)-\rho_{LZ}(Zip(X, Y)) \quad (2)$$

$$I(X : Y) = \rho_{LZ}(X) + \rho_{LZ}(Y) - \rho_{LZCrossed}(X, Y) \quad (3)$$

$$I(X : Y) = \rho_{LZ}(X) - \rho_{LZCrossed}(X|Y) \quad (4)$$

B. k-balanced Alphabets and Sequences

In order to test our sequences we decided to use a sequence that was not as difficult to compress as normal sequences, and that gave us variably difficulty in compression. Thus we chose to use k-balanced sequences.

k-balanced Alphabets are defined as the set of all binary sequences such that they have k "0"s and k "1"s. For example the 1 balanced alphabet is the set $\{01, 10\}$ and the two-balanced alphabet is the set $\{0101, 0110, 0011, 1001, 1010, 1100\}$. The n length k-balanced sequence is the concatenation of the k-balanced alphabet repeated until we have a sequence of length n. For example the length 21 2-balanced sequence is 110010100101001111001.

After some testing we decided that 2-balanced sequences provided the most reasonable view of the compression curve while also allowing our compression to converge in a reasonable amount of time.

C. Methods

In order to test our definitions we used the following algorithm:
1. Begin with 11 empty sequences.
2. Add the 96 bits of the 2-balanced sequence (the alphabet repeated 4 times) to each sequence. 3. For $0 \leq n \leq 10${ For the n^{th} sequence run through the last 96 bits and with probability $n \times 0.1$ flip each bit}.
4. Note that we now have a set of 2-balanced sequences that have $0\%, 10\%, 20\%, \cdots 100\%$ of their bits flipped. For each definition of LZ Mutual Information, run the definition on all of these strings comparing them with the original string (0% chance of flipping) and record the output value.
5. Repeat steps 2-4 until the compression stabilizes.

This algorithm gives us data on the mutual information of sequences where the percent of different bits is known as well as how the sequence compresses over different lengths.

D. Results and Analysis

To analysis our results we piloted the values of each definition and comparison string vs the length of the sequences. This gave us the overall curve of the compression as well as any bounds which this definition seemed to adhere to. For a good definition of Mutual information our values should converge to something between 0 and 1. You will see from our data that this was often not the case for our definitions.

We also plotted the last value of each series vs the percent chance of flipping bits in the strings. This gave us and idea of how close each definition came to actually representing the Mutual Information of the sequences. Originally we though this should be a fairly linear relation; however realized that if you flip all the bits of a k-balanced alphabet you get that k-balanced alphabet in a different order. Thus the parabolic nature of some of the curves may actually be a better result than a completely linear one. More

testing on a different set of sequences is required to verify this though.

Note: For the keys of all vs sequence length graphs each series is defined by the Definition (n) and the percent probiblty of bit flipping in its comparison string (p). They are listed in the notation Definition $< n > (s < p >)$

Definitions 1 and 2 seemed to converge to points between 0.5 and -2 leading us to believe that these definitions of joint compression may have yielded too large of values. We also noted that Definition 2 seems to be bounded by 0 on the top we believe this to be an artifact of running LZ on a zipped value but this will require further testing (See Figures 2 and 2).

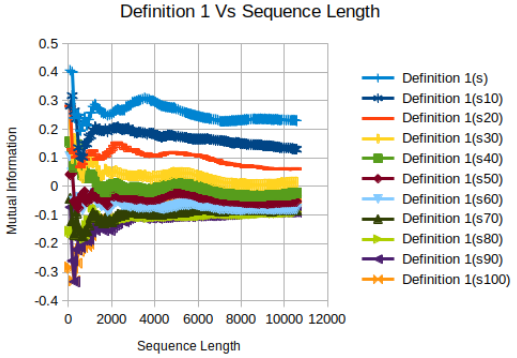

FIG. 2: Mutual Information Vs Sequence Length for Definition 1

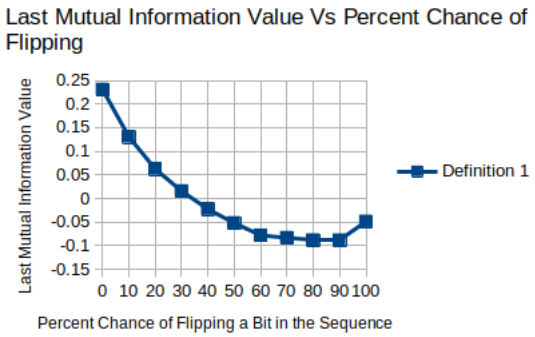

FIG. 3: Last Mutual Information value for Definition 1 vs Percent chance of flipping a bit in the sequence

Other than the bounds problems we are hopeful, definitions 1 and 2 as definitions of Mutual

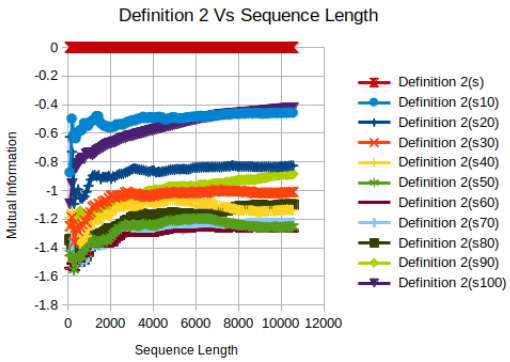

FIG. 4: Mutual Information Vs Sequence Length for Definition 2

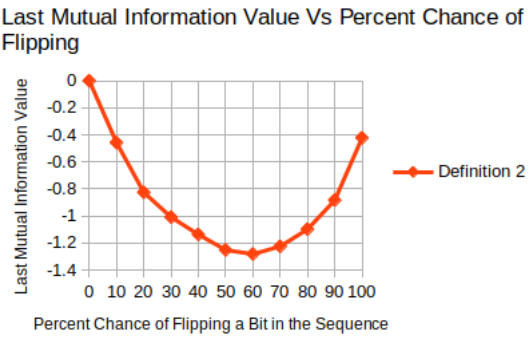

FIG. 5: Last Mutual Information value for Definition 2 vs Percent chance of flipping a bit in the sequence

Information. While their converged mutual information vs percent difference graphs did not display the originally expected linear shape, the parabolic shape that they displayed may have some merit to it that we plan on investigating further (See figures Figures 3 and 5).

Definition 3 was the outlying definition in many ways. First of all it was the only definition to conform to the 0 to 1 bounds we where expecting (See Figure 6). Its Mutual information to percent chance of flipping seemed to be parabolic; however it was the opposite concavity to all other definitions. We are still unsure of why this is and will continue to investigate it (See Figure 7).

Finally we have Definition 4. This definition also broke the 0 to 1 bounds rule we had expected (See Figure 8) and its MI vs percent

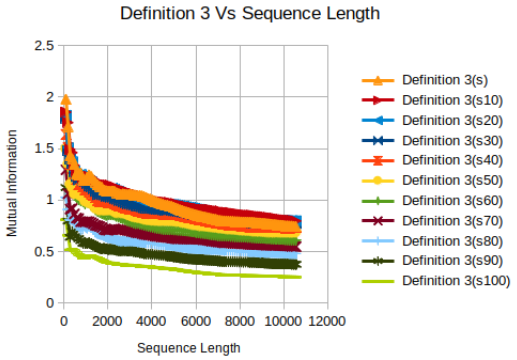

FIG. 6: Mutual Information Vs Sequence Length for Definition 3

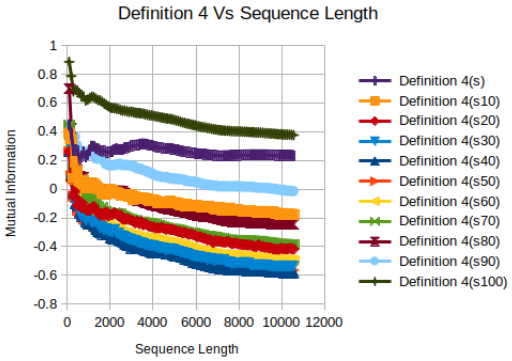

FIG. 8: Mutual Information Vs Sequence Length for Definition 4

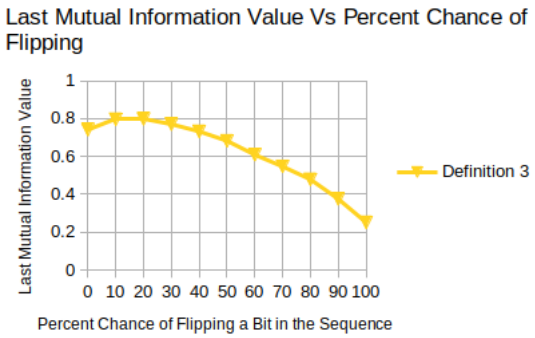

FIG. 7: Last Mutual Information value for Definition 3 vs Percent chance of flipping a bit in the sequence

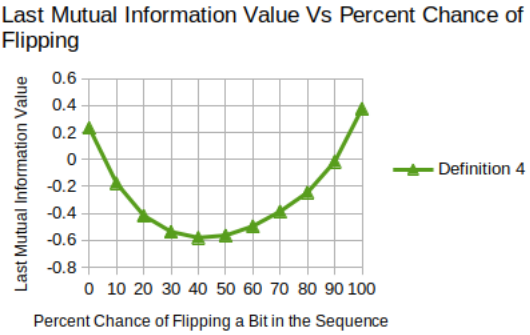

FIG. 9: Last Mutual Information value for Definition 4 vs Percent chance of flipping a bit in the sequence

flipped seems to be a parabola bottoming out at the 40 percent mark which was not at all predicted(See Figure 9).

E. Application to Protein Sequence Alignment

In Bioinformatics we often want to know how related two protein sequences are. To do this we use various algorithms to generate alignments between two or more sequences. These alignments line up each protein to another protein or gap in the other sequence, trying to match as many of the proteins as possible and mitigate gaps. We often compare how related two proteins are by taking the number of exact matches in an alignment divided by the length of the alignment. This is known as the Percent Identity.

Note that we in this case we are trying to find how related two objects are which is very similar to the mutual information. Thus we decided to see how our definitions of mutual information compared to an accepted algorithm with some actual data. To do this we took Alpha Tubulin protein sequences from several different species and got their sequence alignments and percent identities from an alignment tool called Clustal Omega [9]. We then took the Mutual Information for each pair of sequences using our LZ definitions and compared the two.

From our results Definitions 1 and 4 where the only ones that seemed to have any correlation to the Clustal Percent Identity values. In Figure 10 we can see the graph of Mutual Information vs Clustal percent identity. We then created trend lines for both definitions and

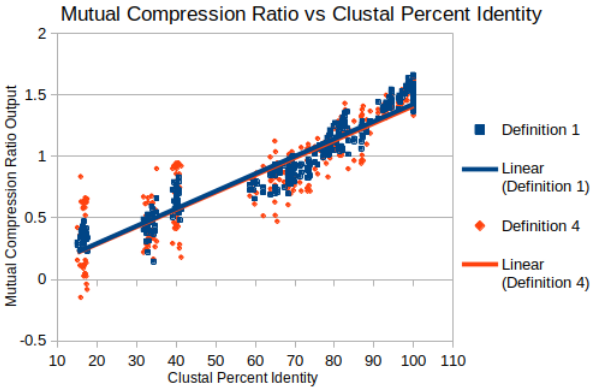

FIG. 10: Shows the Mutual Information Calculated by Definitions 1 and 4 vs the Percent Identity calculated by the Clustal Omega algorithm

got the R^2 values. For Definition 1 the trend line R^2 value was 0.913 and for Definition 2 it was 0.802, neither of which are enough to say they where highly correlated but it is certainly enough to warrant further exploration.

III. THE ADVENTURE CONTINUES

In the future we plan to continue studying an improving our definitions of LZ Mutual Information. In particular we have created a worst case scenario for LZ compression (See Section III A) and plan to use this to redefine our definition of the LZ compression ratio such that we are taking the length of the LZ encoding over the length of the LZ encoding of the "Worst" sequence. This will keep the compression ratio between 0 and 1 and hopefully stabilize our definitions between these bounds as well. We also plan to use it to make more mathematically rigorous proofs about these definitions.

A. The Worst Sequence

In order to further study LZ we came up with a worst case scenario i.e. the least compressible binary sequence for LZ. In order to always get the worst case scenario we need to maximize the size of the dictionary. This both keeps the algorithm from being able to encode large sections of the sequence with a single key word and it also increases the size of the key words. In order to maximize the dictionary we need to make sure that every possible binary string of increasing size are in the dictionary. This does work for many definitions or LZ; however remember that our definition actually has a look ahead property that allows it to compress a little more efficiently and it starts out with the alphabet in its dictionary so adding a single 0 and 1 is not helpful.

Thus we propose a sequence that instead we concatenate all binary words of increasing length (beginning with length 2) such that the last letter of each word is used in the subsequent word. For example if we take the binary strings of length 2 (00,01,10,11) let us start with 10, we then concatenate 00 to it using the already existing 0 to create 100. Next we add 01 to get 1001 and finally we add 11 to get the sequence 10011. Note that all binary sequences of length 2 are sub-sequences of the sequence we created and are compressed into as sequence of length 5. We can see a similar process for the sequences of length 3 in Table III. Note that any set of binary sequences of length n will have an equal number of sequences that start and end in 0 and 1 meaning that we can repeat this possess infinitely to make an infinitely long sequence.

$$1\,0\,0\,1\,\overline{1\,0\,0}\,\overline{0\,0\,1}\,\overline{0\,1\,1}\,\overline{1\,0\,1}\,\overline{0\,1\,1}\,1\,1\ldots$$

TABLE III: Depicts how the binary sequences of length 3 are encoded into the Worst Sequence

Note that using this algorithm we can encode all binary sequences of length n into a sequence that is length $n \geq 2\,n + (n-1)(2^n - 1)$ because there we begin with a sequence of length n and then concatenate $(2^n - 1)$ sequences (number of binary sequences of length n minus the one we already used) each of length n-1 (length minus the one bit we use to prepend) giving us a total length of $n \geq 2\,n + (n-1)(2^n - 1)$ bits. Note that the overall sequence generated by this algorithm includes all binary sequences of length $2 \leq n \leq N$ in $5 + \sum_{n=3}^{N} n + (n-1)(2^n - 1) - 1$ bits. This

is because we start with 5 bits for 2 and then add $n + (n-1)(2^n - 1) - 1$ for each new set of sequences of length n minus one each time for the prepend.

00	→ 01	000 → 001		010 → 011	
↑	↓	↑	↓	↑	↓
10	11 → 100		101 → 110		111 → ⋯

TABLE IV: Flow of LZ dictionary creation when compressing the Worst sequence.

This also maximizes the number of words LZ puts into its dictionary and therefore it maximizes the encoding generated by LZ. See Table IV to see the flow of dictionary creation for LZ [10].

IV. ACKNOWLEDGEMENTS AND REFERENCES

I would like to acknowledge Dr. Adam Case for being my advisor for the CS side of this project. Also for his creation of k-balanced sequences that where used in the testing stage of this project (See Section II B). I would also like to acknowledge Dr. Jerry Honts for being my Advisor for the Biology side of this project and for putting together a set of proteins that had both a wide range of conservation and where long enough for LZ to run efficiently.

[1] Britannica, T. Editors of Encyclopaedia (2022, December 7). GIF. Encyclopedia Britannica. https://www.britannica.com/technology/GIF

[2] ZIV, JACOB, and ABRAHAM LEMPEL. "A Universal Algorithm for Sequential Data Compression - Duke University." Duke University, TRANSACTIONS ON INFORMATION THEORY, 1977, https://courses.cs.duke.edu/spring03/cps296.5/papers/ziv_lempel_1977_universal_algorithm.pdf.

[3] Ziv, Jacob, and Abraham Lempel. "Compression of Individual Sequences via Variable-Rate Coding." Compression of Individual Sequences via Variable-Rate Coding, Duke University, 1978, https://courses.cs.duke.edu/spring03/cps296.5/papers/ziv_lempel_1978_variable-rate.pdf

[4] Welch, Terry. "A Technique for High-Performance Data Compression - Duke University." a Technique for High-Performance Data Compression, Duke University, 1984, https://courses.cs.duke.edu/spring03/cps296.5/papers/welch_1984_technique_for.pdf

[5] Shannon, Claude E., and Warren Weaver. The Mathematical Theory of Communication. The Bell System Technical Journa, 1948. https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

[6] Fano, Robert M. "A Measure of Information." Transmission of Information: A Statistical Theory of Communication, Mit Press, S.l., 2003.

[7] Rita, Luis. Normalized Mutual Information. 6 May 2020, https://luisdrita.com/normalized-mutual-information-a10785ba4898.

[8] Cover, T. M., and Joy A. Thomas. Elements of Information Theory. Second ed., Wiley, 2012.

[9] Madeira F, Pearce M, Tivey ARN, et al. Search and sequence analysis tools services from EMBL-EBI in 2022. Nucleic Acids Research. 2022 Apr:gkac240. DOI: 10.1093/nar/gkac240. PMID: 35412617; PMCID: PMC9252731.

[10] Shor, Peter. "Lempel-Ziv Notes Prof. Peter Shor - Math.mit.edu." Lempel-Ziv Notes, MIT, https://math.mit.edu/~djk/18.310/Lecture-Notes/LZ-worst-case.pdf.