2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Social Network Analysis on Interpretable Compressed Sparse Networks

Connor C.J. Hryhoruk
Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada

Carson K. Leung(✉)
Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada
Carson.Leung@UManitoba.ca

***Abstract*—Big data are everywhere. World Wide Web is an example of these big data. It has become a vast data production and consumption platform, at which threads of data evolve from multiple devices, by different human interactions, over worldwide locations, under divergent distributed settings. Embedded in these big web data is implicit, previously unknown and potentially useful information and knowledge that awaited to be discovered. This calls for web intelligence solutions, which make good use of data science and data mining (especially, web mining or social network mining) to discover useful knowledge and important information from the web. As a web mining task, web structure mining aims to examine incoming and outgoing links on web pages and make recommendations of frequently referenced web pages to web surfers. As another web mining task, web usage mining aims to examine web surfer patterns and make recommendations of frequently visited pages to web surfers. While the size of the web is huge, the connection among all web pages may be sparse. In other words, the number of vertex nodes (i.e., web pages) on the web is huge, the number of directed edges (i.e., incoming and outgoing hyperlinks between web pages) may be small. This leads to a sparse web. In this paper, we present a solution for interpretable mining of frequent patterns from sparse web. In particular, we represent web structure and usage information by bitmaps to capture connections to web pages. Due to the sparsity of the web, we compress the bitmaps, and use them in mining influential patterns (e.g., popular web pages). For explainability of the mining process, we ensure the compressed bitmaps are interpretable. Evaluation on real-life web data demonstrates the effectiveness, interpretability and practicality of our solution for interpretable mining of influential patterns from sparse web.**

***Index Terms*—Web intelligence, Intelligent agent technology, World wide web, Web of data, Data science, Data analytics, Data mining, Frequent pattern mining, Web mining, Web structure mining, Web usage mining, Data compression, Bitmap, Recommendation, Explainability, Interpretability**

I. Introduction

Web has evolved as an omnipresent system, which highly influences education, industry, science, and our daily life. Web has become a vast data production and consumption platform, at which threads of data evolve from multiple devices, by different human interactions, over worldwide locations, under divergent distributed settings. Such a dynamic complex system demands adaptive intelligent solutions, which advance knowledge, human interactions and innovation. This calls for *web intelligence* to address issues towards deepening the understanding of entities, phenomena, and developments on the web.

IEEE/ACM ASONAM 2022, November 10-13, 2022
978-1-6654-5661-6/22/$31.00 © 2022 IEEE

Web intelligence solutions generally make good use of data science, artificial intelligence (AI) [1–3], data mining [4–8], data analytics [9, 10], mathematics, statistics, informatics, and/or visual analytics [11–15]. For instance, a web intelligence solution can apply AI to examine data from a connected world [16] of webs—including web of agents [17, 18], data [19], people [20–23], things [24–26], and trust [27]—for social good. As a second instance, another web intelligence solution can apply data mining—in particular, web mining—to examine these big web data for knowledge discovery.

Web mining can be further subdivided into three common mining tasks such as the mining of web contents, structures, and usage. Among them:

- Web content mining focuses on examining the contents of the web pages. These include useful data, information and knowledge on the web pages.
- Web structure mining focuses on examining the structure of the web. These include vertex nodes (i.e., web pages), edges (i.e., hyperlinks—including incoming and outgoing links—between web pages), and their relationships and semantics.
- Web usage mining focuses on examining the surfer logs and their surfing behaviors and patterns. Based on the mined patterns, one can make recommendations of popular and/or influential web pages.

In this paper, we focus on the latter two—i.e., *web structure mining* and *web usage mining*. Specifically, we present a web intelligence solution for interpretable mining of influential patterns from sparse web. This work is motivated by related works in the past few editions of ACM/IEEE ASONAM conferences. To elaborate, a serial web usage mining algorithm [28] was presented in ASONAM-FAB 2016. It represents a world wide web as a collection of bitmaps, in which each bitmap captures outgoing connections of a web page on the web. As such, a directory page may have a high number of outgoing hyperlinks (i.e., out-links for short) pointing to many other web pages. In duality, an influential web page may have a high number of incoming hyperlinks (i.e., in-links for short). Equivalently, such an influential web page would appear in many of the bitmaps (i.e., such an influential web page would be referenced by out-links of many referencing web pages). Frequent pattern mining on these out-links would discover these influential web pages. Note that, with a huge number of web pages on the world wide web, not all web pages would be referenced by every web pages. Many web pages are often referenced by a certain number of web pages. In other words, in terms of density, it is a *(very) sparse* web.

To speed up the web usage mining process, a parallel web usage mining algorithm [29] was presented in ASONAM-FAB 2017. Along this direction, to further reduce the memory space required for capturing the web usage, web data compression [30] was presented in ASONAM-FAB 2018. While the compressed bitmaps were effectively used in web usage mining for frequent, the compressed bitmaps may not be easily comprehensible. Due to the popularity of explainable AI (XAI), an explainable representation of compressed bitmaps [31] was presented in ASONAM-FAB 2020. A flexible representation [32] was presented in ASONAM-FAB 2021 to allow flexibility in compressing groups of 31 consecutive zeros and few 1s in the preceding (or succeeding 32 bits) in a bitmap. However, depending on their distribution, these few 1s sometimes are distributed beyond 32 bits but within 64 or 96 bits. A logical question is: Can we handle a few 1s spread over 64 or 96 bits preceding (or succeeding) groups of 31 consecutive zeros?

In response, we present a solution to this question. *Key contributions* of this paper is our design and development of the interpretable mining of influential patterns from sparse web. Each of our compressed bitmaps effectively captures its out-links to its referencing pages. Each bitmap can also be easily interpreted, which greatly improve explainability of the mining process.

We organize the remainder of this paper as follows. The next section discusses background and related works. Section III describes our interpretable compressed bitmaps for supporting both the mining of influential patterns from sparse web and the recommendation of influential web pages. Evaluation and conclusions are presented in Sections IV and V, respectively.

II. Background and Related Works

A common way to represent and compress a bitmap is to use word-align hybrid (WAH) and position-list WAH (PLWAH) compressed models. These models have been adapted for web mining.

A. Word-Align Hybrid (WAH) Model

To elaborate, the WAH [33] compresses a bitmap by dividing it into groups of 31 bits of 0s or 1s. A group (of 31 bits) can be:

- all 0s (i.e., a zero-filled group);
- a mixture of 0s and 1s (i.e., a literal word); or
- very rarely, all 1s (i.e., a one-filled group).

Let us focus on the first two options because they are more common. Consecutive groups of zero-filled words can form a zero-filled word, in which the bitmap captures the number of these consecutive zero-filled groups. See Example 1.

Example 1. *Consider a web page (say, web page 1) that references web pages 9, 31759, 63365, 63366, 63368, 63372, 109926, 109941 and 109957. The original bitmap to capture these referenced pages would take at least 109,926 bits. In this bitmap, all except seven bits (representing these seven referenced pages) are 0s. With the WAH compression model, the resulting WAH compressed bitmap becomes:*

1) *[0] [00000 00010 00000 00000 00000 00000 0]*
2) *[10] [00 0000 0000 0000 0000 0011 1111 1111]*
3) *[0] [00000 00000 00001 00000 00000 00000 0]*
4) *[10] [00 0000 0000 0000 0000 0011 1111 1011]*
5) *[0] [11010 00100 00000 00000 00000 00000 0]*
6) *[10] [00 0000 0000 0000 0000 0101 1101 1100]*
7) *[0] [00000 00000 00000 00000 00000 00000 1]*
8) *[0] [00000 00000 00001 00000 00000 00000 1]*

Here, web pages 9, 31759, 63365, 63366, 63368, 63372, 109926, 109941 and 109957 are represented by 1s in literal words 1, 3, 5, 5, 5, 5, 7, 8 and 8. These literal words are prefixed by [0]. The gaps between these literal words are zero-filled words 2, 4 and 6 (which are prefixed by [10]). The binary values in the suffix of these zero-filled words indicate the number of consecutive groups of 31 zeros. To elaborate, the 1 at the 9th position of literal word 1 represents web page 9. This literal word is succeeded by $11\ 1111\ 1111_{(2)} = 1023_{(10)}$ *groups of 31 consecutive 0s (as indicated by the binary value in the zero-filled word 2), and the 1 at the 15th position of literal word 3 represents web page* $31759 = 31 + (1023 \times 31) + 15$*. This literal word is succeeded by* $11\ 1111\ 1011_{(2)} = 1019_{(10)}$ *groups of 31 consecutive 0s (as indicated by the binary value in the zero-filled word 4), and the 1s at the 1st, 2nd, 4th and 8th positions of literal word 5 represent web pages 63365, 63366, 63368,* $63372 = 31 + (1023 \times 31) + 31 + (1019 \times 31) + \{1 — 2 — 4 — 8\}$*. This literal word is succeeded by* $101\ 1101\ 1100_{(2)} = 1500_{(10)}$ *groups of 31 consecutive 0s (as indicated by the binary value in the zero-filled word 6), and the 1 at the 31st position of literal word 7 represents web page* $109926 = 31 + (1023 \times 31) + 31 + (1019 \times 31) + 31 + (1500 \times 31) + 31$*. Finally, 1s at the 15th and 31st position of literal word 8 represent web page 109941 and* $109957 = 31 + (1023 \times 31) + 31 + (1019 \times 31) + 31 + (1500 \times 31) + 31 + \{15 — 31\}$*. Note that this WAH compressed bitmap requires only* 8×32 *bits = 256 bits.*

B. Position-List WAH (PLWAH) Model

With 30 bits in the suffix of a zero-filled word, $2^{30} - 1 >$ 1 billion consecutive groups of 31 zeros can be represented. Most often, gaps between literal words are shorter than 1 billion consecutive groups of 31 zeros. As such, PLWAH [33] takes advantage of the used space in this 30-bit suffix by capturing positions of a single 1 succeeding consecutive groups of 31 zeros. See Example 2.

Example 2. *Continue with the previous example. A PLWAH compressed bitmap becomes:*

1) *[0] [00000 00010 00000 00000 00000 00000 0]*
2) *[10] [01111] [0 0000 0000 0000 0011 1111 1111]*

3) *[10] [00000] [0 0000 0000 0000 0011 1111 1011]*
4) *[0] [11010 00100 00000 00000 00000 00000 0]*
5) *[10] [11111] [0 0000 0000 0000 0101 1101 1100]*
6) *[0] [00000 00000 00001 00000 00000 00000 1]*

Note that the single 1 in literal word 3 (i.e., web page 31759) in the WAH compressed bitmap is now merged with the zero-filled word 2. The position of that 1 (i.e., 15th position) is captured by [01111]. Similarly, the single 1 in literal word 7 (i.e., web page 109926) in the WAH compressed bitmap is now merged with its preceding zero-filled word, and its position (i.e., 31st position) is captured by [11111]. This PLWAH compressed bitmap requires only 6 $\times$ 32 bits = 192 bits.

C. Compact Bitwise Representation for Web Mining (CBWeb) Model

The CBWeb [30] adapts the WAH and PLWAH models for web mining. Observing the availability of space in the suffix of zero-filled word for capturing more than a single 1 succeeding consecutive groups of 31 zeros, CBWeb captures a few 1s succeeding consecutive groups of 31 zeros and uses the space to indicate the positions of these few 1s. See Example 3.

Example 3. *Continue with the previous example. A CBWeb(4) compressed bitmap—which captures at most four 1s succeeding consecutive groups of 31 zeros—becomes:*

1) *[0] [00000 00010 00000 00000 00000 00000 0]*
2) *[10] [01111] [00000] [00000] [00000] [11 1111 1111]*
3) *[10] [00001] [00010] [00100] [01000] [11 1111 1011]*
4) *[10] [00000] [00000] [00000] [00000] [11 1111 1111]*
5) *[10] [11111] [00000] [00000] [00000] [01 1101 1101]*
6) *[0] [00000 00000 00001 00000 00000 00000 1]*

Note that the four 1s in literal word 4 (i.e., web pages 63365, 63366, 63368 and 63372) in the PLWAH compressed bitmap is now merged with the zero-filled word 3. Positions of these 1s are captured by [00001], [00010], [00100] and [01000]. This CBWeb(4) compressed bitmap requires 6 $\times$ 32 bits = 192 bits.

D. Explainable CBWeb (XCBWeb) Model

In a CBWeb zero-filled word, positions of a few 1s come before the number of consecutive groups of 0s. However, these few 1s are succeeding the consecutive groups of 0s. Hence, such an order can be confusing, Observing that this ordering of information within each zero-filled word in the CBWeb compressed bitmap may be not too intuitive, XCBWeb [31] rearranges the bits within each word of the bitmap so that information can be logically read from left to right. See Example 4.

Example 4. *Continue with the previous example. A XCBWeb(4) compressed bitmap becomes:*

1) *[0] [00000 00010 00000 00000 00000 00000 0]*
2) *[10] [11 1111 1111] [01111] [00000] [00000] [00000]*
3) *[10] [11 1111 1011] [00001] [00010] [00100] [01000]*
4) *[10] [11 1111 1111] [00000] [00000] [00000] [00000]*
5) *[10] [01 1101 1101] [11111] [00000] [00000] [00000]*
6) *[0] [00000 00000 00001 00000 00000 00000 1]*

Note that bits within each of zero-filled words 2, 3, 4 and 5 are rearranged. For instance, zero-filled word 2 now represents $[11\ 1111\ 1111]_{(2)} = 1023_{(10)}$ groups of 31 consecutive 0s are succeeded by a literal word with a single 1 in its $[01111]_{(2)}$ = 15th position. This literal word is then succeeded by zero-filled word 3 representing $[11\ 1111\ 1011]_{(2)} = 1019_{(10)}$ groups of 31 consecutive 0s, which are succeeded by a literal word with four 1s. This literal word is then succeeded by zero-filled words 4 and 5 representing $[11\ 1111\ 1111]_{(2)} + [01\ 1101\ 1101]_{(2)} = 1500_{(10)}$ groups of 31 consecutive 0s, which are succeeded by a literal word with a single 1 at its $[11111]_{(2)}$ = 31st position.

E. Flexible XCBWeb (FXCBWeb) Model

Observed from the XCBWeb(k) bitmap (where k=4) in Example 4 that the bitmap starts with a literal word 1. Hence, it cannot be merge with any zero-filled word (because there is not preceding zero-filled word). We also observed that there are insufficient bits to represent large groups (e.g., > 1023 groups) of 31 consecutive zeros. Hence, it splits into two zero-filled words 4 and 5. These two observations can be generalized to become the following: (a) If a bitmap starts with a literal word, it cannot be compressed with the second word. (b) For large k, there may be insufficient bits to represent large groups of 31 consecutive zeros.

In response to these two observations and address the problems associated with these observations, FXCBWeb allows the flexibility to have the usual arrangement of having groups of 31 consecutive zeros followed by a few 1s, and a new additional arrangement of having a few 1s followed by groups of 31 consecutive zeros. Because of this flexibility, it uses the new arrangement to capture a few 1s before a split portion of large groups of 31 consecutive zeros, and the remaining portion can be captured by the usual arrangement. As a preview, in the next example, the 1500 groups of 31 consecutive zeros were (a) preceded by the four 1s at 63365th, 63366th, 63368th and 63372nd positions and (b) succeeded by a single 1 at the 31st position. When implementing this flexibility, it uses the second bit of a zero-filled word to indicate which arrangement is used for the zero-filled word. Specifically:

- *prefix [10] indicates the usual arrangement*, with which we put the groups of 31 consecutive zeros before the positions of the few 1s and
- *prefix [11] indicates the new arrangement*, with which we put the positions of the few 1s before the groups of 31 consecutive zeros.

Example 5. *Continue with the previous example. A FXCBWeb(4) compressed bitmap becomes:*

1) *[11] [01001] [00000] [00000] [00000] [11 1111 1111]*
2) *[11] [01111] [00000] [00000] [00000] [11 1111 1011]*
3) *[11] [00001] [00010] [00100] [01000] [11 1111 1111]*
4) *[10] [01 1101 1101] [11111] [00000] [00000] [00000]*
5) *[0] [00000 00000 00001 00000 00000 00000 1]*

Note that, while FXCBWeb(4) compressed bitmap in this example captures the same information as the XCBWeb(4) compressed bitmap in the previous example, FXCBWeb(4) compressed bitmap requires less space—only requires 5 $\times$ 32 bits = 160 bits.

Let us explain each of the four zero-filled words in this FXCBWeb(4) compressed bitmap in details:

- *The prefix [11] of zero-filled word 1 indicates that it uses the new arrangement, which captures the position of a 1 (i.e., at [01001]$_{(2)}$ = 9th position for web page 9) before [11 1111 1111]$_{(2)}$ = 1023$_{(10)}$ groups of 31 consecutive zeros. This word in the FXCBWeb(4) bitmap captures literal word 1 and the first half of zero-filled word 2 in the XCBWeb(4) bitmap. This gives an example of a bitmap starting with a literal word and compressing with the second word. In other words,* for a bitmap starts with a literal word, it can now be compressed with the second word.
- *It is succeeded by zero-filled word 2, which captures the position of a 1 (i.e., at [01111]$_{(2)}$ = 16th position for web page 31759) before [11 1111 1011]$_{(2)}$ = 1019$_{(10)}$ groups of 31 consecutive zeros. This word in the FXCBWeb(4) bitmap captures the second half of zero-filled word 2 and the first half of zero-filled word 3 in the XCBWeb(4) bitmap.*
- *It is then succeeded by zero-filled word 3, which captures positions of four 1s (at [00001]$_{(2)}$ = 1st, [00010]$_{(2)}$ = 2nd, [00100]$_{(2)}$ = 4th and [01000]$_{(2)}$ = 16th positions representing web pages 63365, 63366, 63368 and 63372) before [11 1111 1111]$_{(2)}$ = 1023$_{(10)}$ groups of 31 consecutive zeros. This word in the FXCBWeb(4) bitmap captures the second half of zero-filled word 3 and the first half of zero-filled word 4 in the XCBWeb(4) bitmap.*
- *The prefix [10] of zero-filled word 4 indicates that it uses the usual arrangement, which captures [01 1101 1101]$_{(2)}$ = 477$_{(10)}$ groups of 31 consecutive zeros first. Recall that the previous word 3 ends with 1023 groups of 31 consecutive zeros. When combined it with the current 477 groups, they give a total of 1500 groups of 31 consecutive zeros. This is an example of* representing large groups (e.g., $>$ 1023 groups) of 31 consecutive zeros. *These groups of 31 consecutive zeros are then succeeded by the position of a 1 (i.e., at [11111]$_{(2)}$ = 31st position for web page 109926).*
- *Finally, the literal word 5 cannot be compressed by FXCBWeb(4).*

This example illustrates how FXCBWeb compressed bitmap successfully addresses the two aforementioned concerns. Such a compressed bitmap provides an interpretable representation.

III. Mining with Our Multi-Word (MWXCBWeb) Model

Observed from Example 5 that, while FXCBWeb(4) helps reduce the size of bitmaps by providing flexible compression, the resulting bitmap still requires 5 $\times$ 32 bits = 160 bits. This is due to the two 1s in the last literal word are located beyond the 31 bits succeeding the groups of 31 consecutive zeros in the zero-filled word 4. To elaborate, web pages 63366, 63368 and 63372 span beyond 31 bits. If they were spanned within the same 31 bits preceding or succeeding the groups of 31 consecutive zeros, then they can be compressed by FXCBWeb(4) and their positions can be captured by a zero-filled word as we did for web pages 63365, 63366, 63368 and 63372 (which span within 31 bits).

A. Our Multi-Word (MWXCBWeb) Model

To deal with the aforementioned situation where a few 1s span over some tiny multiple of 31 bits (e.g., 31, 62, 93 bits), we present our multi-word explainable CBWeb (MWXCBWeb) model. In general, a zero-fill word in MWXCBWeb(k)—for $1 \leq k \leq 4$ practically (though $1 \leq k \leq 5$ theoretically)—is represented as a 32 bit word with:

- prefix "10" to indicate the usual arrangement (i.e., putting the groups of 31 consecutive zeros before the positions of the few 1s) in a zero-filled word; prefix "11" to indicate the flexible arrangement as in FXCBWeb (i.e., putting the positions of the few 1s before the groups of 31 consecutive zeros)
- Positions of the few 1s are represented by 5 bits;
- In *between* each of these 5-bit positions, we use an extra bit to flag/indicate whether the succeeding position is on the same 31-bit word as its preceding position. A "0" bit indicates the next ?1?-bit is on the same word as the current one; an "1" bit indicates the next ?1?-bit is on the next word. By doing so, the maximum of $k = 4$ "1" bits can span within the same 31-bit word (with these $k - 1 = 3$ flags/indicator bits all "0") or over $k \times 31$ = 124 bits (with these 3 flags/indicator bits all "1")

Let us illustrate the key ideas by continuing with Example 5.

Example 6. *Continue with the previous example. A MWXCBWeb(3) compressed bitmap becomes:*

1) *[11] [01001] [0] [00000] [0] [00000] [0 0011 1111 1111]*
2) *[11] [01111] [0] [00000] [0] [00000] [0 0011 1111 1011]*
3) *[11] [00001] [0] [00010] [0] [00100] [0 0011 1111 1111]*
4) *[10] [0 0101 1101] [11111] [1] [01111] [0] [11111]*

Note that, while MWXCBWeb(3) compressed bitmap in this example captures the same information as the FXCBWeb(3) compressed bitmap in the previous example, MWXCBWeb(3) compressed bitmap requires less space—only requires 4 $\times$ 32 bits = 128 bits.

Let us explain each of the four zero-filled words in this MWXCBWeb(3) compressed bitmap in details:

- *The prefix [11] of zero-filled word 1 indicates that it uses the new arrangement, which captures the position of a 1 (i.e., at [01001]$_{(2)}$ = 9th position for web page 9) before [11 1111 1111]$_{(2)}$ = 1023$_{(10)}$ groups of 31 consecutive zeros. This word in the MWXCBWeb(3) bitmap captures literal word 1 and the first half of zero-filled word 2 in the*

XCBWeb(3) bitmap. This gives an example of a bitmap starting with a literal word and compressing with the second word. In other words, for a bitmap starts with a literal word, it can now be compressed with the second word.

- *It is succeeded by zero-filled word 2, which captures the position of a 1 (i.e., at [01111]$_{(2)}$ = 16th position for web page 31759) before [11 1111 1011]$_{(2)}$ = 1019$_{(10)}$ groups of 31 consecutive zeros. This word in the MWXCBWeb(3) bitmap captures the second half of zero-filled word 2 and the first half of zero-filled word 3 in the XCBWeb(3) bitmap.*
- *It is then succeeded by zero-filled word 3, which captures positions of three 1s before [11 1111 1111]$_{(2)}$ = 1023$_{(10)}$ groups of 31 consecutive zeros. This word in the MWXCBWeb(3) bitmap captures the second half of zero-filled word 3 and the first half of zero-filled word 4 in the XCBWeb(3) bitmap. Note that the first 1 is at [00001]$_{(2)}$ = 1st position representing web page 63365. The first flag [0]—located between the first and second 5-bit positions for the corresponding 1s indicates the next position is on the word, and this second 1 is at [00010]$_{(2)}$ = 2nd position representing web page 63366. The second flag [0] indicates the next position is on the same word, and this second 1 is at [00010]$_{(2)}$ = 2nd position representing web page 63366. The second flag [0]—located between the second and third 5-bit positions for the corresponding 1s indicates the next position is on the word, and this third 1 is at [00100]$_{(4)}$ = 4th position representing web page 63366.*
- *The prefix [10] of zero-filled word 4 indicates that it uses the usual arrangement, which captures [01 1101 1101]$_{(2)}$ = 477$_{(10)}$ groups of 31 consecutive zeros first. Recall that the previous word 3 ends with 1023 groups of 31 consecutive zeros. When combined it with the current 477 groups, they give a total of 1500 groups of 31 consecutive zeros. This is an example of* representing large groups (e.g., $>$ 1023 groups) of 31 consecutive zeros. *These groups of 31 consecutive zeros are then succeeded by the position of three 1s. The first 1 is at [11111]$_{(2)}$ = 31st position for web page 109926. Then, the first flag [1] in between the first and second 1s indicates the second 1 (representing the next web page) is on the* following *word from the first 1. More precisely, [1] [01111]$_{(2)}$ = 15th position on the next word for web page 109941. Similarly, the second flag [0] in between the second and theird 1s indicates the third 1 (representing the next web page) is on the* same *word as the second 1. More precisely, [0] [11111]$_{(2)}$ = 31st position on the same word for web page 109957. 477$_{(10)}$*

This example illustrates how MWXCBWeb compressed bitmap successfully handles a few 1s (e.g., three 1s) spanning over the same or multiple words. Inherited from FXCBWeb(k), our MWXCBWeb(k) is also interpretable.

B. Our Mining and Recommendation of Influential Patterns

With our aforementioned MWXCBWeb compression model, we can represent a bitmap for capturing outgoing connections of a web page on the web. Repeating the compression for other bitmaps, we can then obtain a collection of MWXCBWeb compressed bitmaps capturing outgoing connections of many web pages on the web. Then, we adapt the serial or parallel web mining algorithms that we described in Section 1 to find frequent patterns from the collection of compressed bitmaps.

For *web structure mining*, our web mining algorithms examine the outgoing links on web pages to discover frequently referenced pages. These frequently referenced pages are popular pages. Key ideas of the web mining process can be described as follows. First, we discover those frequently referenced pages as popular or influential web pages to be recommended to users. Then, for each popular web page, we form a projection to examine what other frequently referenced web pages have incoming links from the same referencing pages. We repeat this process in a depth-first manner until no further projection can be form. This iterative process would terminate because each projection results in a search space not larger than that in previous iterations. At the end of this iterative process, we discover sets of frequently referenced web pages. These sets can be recommended to the users as a collection of highly influential web pages, which are frequently referenced by other pages.

Similarly, for *web usage mining*, our web mining algorithms examine the web logs to discover frequently visited, browsed and/or surfed web pages. These frequently visited pages are popular pages. Again, we first discover those frequently visited pages as popular or influential web pages to be recommended to web surfers. We then iteratively discover sets of frequently visited web pages. These sets can be recommended to the web surfers as a collection of highly influential web pages, which are frequently visited by web surfers. Moreover, when a web surfer visits a subset of frequently visited web pages, we can recommend the remainder of the set of frequently visited web pages.

IV. Evaluation

A. Correctness

To evaluate our solution for interpretable mining of influential patterns from a sparse web, we first examined the correctness of our solution. Our MWXCBWeb(k) compressed bitmaps capture the same information (e.g., outgoing links) as existing , FXCBWeb(k), XCBWeb(k), CBWeb(k), PLWAH and WAH compressed bitmaps. Our corresponding mining algorithm returns the same collection of frequently referenced or visited web pages.

B. Functionality

Afterwards, we examined the functionalities of our model. Table I shows that, in terms of **compression**, our MWXCBWeb model:

TABLE I: Evaluation on functionalities

	WAH [33]	PLWAH [33]	CBWeb [30]	XCBWeb [31]	FXCBWeb [32]	**Our MWXCBWeb**
Compress groups of 31 consecutive 0s	√	√	√	√	√	√
Merge groups of 31 consecutive 0s with *a single 1* in a literal word *succeeding* the 0s	×	√	√	√	√	√
Merge groups of 31 consecutive 0s with *a few 1s* in a literal word succeeding the 0s	×	×	√	√	√	√
Order bits in the same logical order as in the original uncompressed bitmaps	×	×	×	√	√	√
Merge groups of 31 consecutive 0s with a few 1s in a literal word *preceding* the 0s	×	×	×	×	√	√
Merge groups of 31 consecutive 0s with *a few 1s* in *multiple* consecutive literal words succeeding the 0s	×	×	×	×	×	√
Merge groups of 31 consecutive 0s with a few 1s in *multiple* consecutive literal words preceding the 0s	×	×	×	×	×	√

(a) compresses groups of 31 consecutive zeros (cf. uncompressed bitmaps);
(b) merges groups of 31 consecutive zeros with a single 1 in a literal word succeeding the groups of zeros (cf. WAH, which cannot merge with any 1 in the literal word); and
(c) groups of 31 consecutive zeros with a few 1s in a literal word succeeding the groups of zeros (cf. PLWAH, which merges with only a single 1 in the literal word).

Then, when examining the functionalities of our model, we also evaluated its **interpretability** in addition to compression. Results show that our MWXCBWeb model:

(d) orders the bits in the same logical order as in the original uncompressed bitmaps (cf. CBWeb, in which positions of 1s in a literal word appear before the binary value of their preceding groups of consecutive zeros).

Moreover, when examining the functionalities of our model, we evaluated its **flexibility** in addition to compression and interpretability. Results show that our MWXCBWeb model:

(e) has the flexibility to merge groups of 31 consecutive zeros with a few 1s in a literal word *preceding* the groups of zeros. In contrast, CBWeb and XCBWeb, which can only merge groups of 31 consecutive zeros with a few 1s in a literal word *succeeding*—but not preceding—the groups of zeros.

To recap, our MWXCBWeb model provides compression, interpretability and flexibility.

C. Space and Time Requirements

We evaluated the compression ratio, memory consumption, and runtime with datasets from from the Stanford Large Network Collection[1] and SuitSparse Matrix Collection[2] of the Stanford Network Analysis Platform (e.g., ego-Gplus, ego-Twitter, web-BerkStan, web-Google, web-NotreDame). Compression ratio measures the size of compressed bitmaps when compared with the original uncompressed bitmaps. Figure 1 shows that our MWXCBWeb(k) model led to the best compression ratio. For instance, the size of the FXCBWeb compressed bitmap for 400 ego-users is less than 10% of the size of uncompressed bitmaps. Due to the compression ratio, Figure 2 shows that our MWXCBWeb(k) consumes the least amount of memory. It, in turn, reduces the runtime due to the small size of bitmaps to be loaded and scanned.

[1]https://snap.stanford.edu/data/

[2]https://sparse.tamu.edu/SNAP

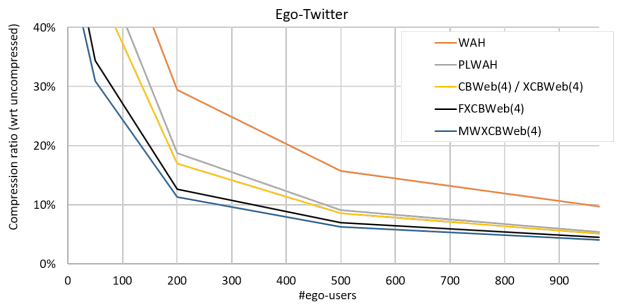

Fig. 1: Compression ratio

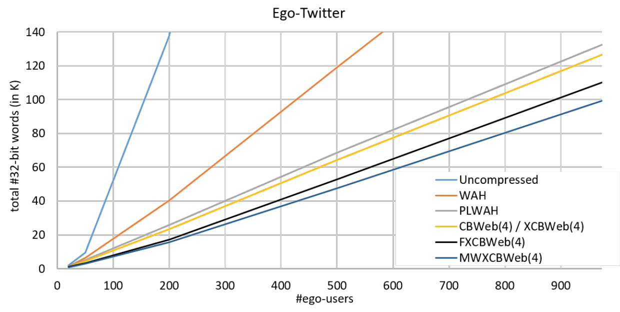

Fig. 2: Memory consumption

V. Conclusions

In this paper, we present a solution for interpretable mining of influential patterns from sparse web. In particular, we represent web structure and usage information (e.g., connections to web pages) by multiple-word, explainable, compact bitwise representation for web mining (MWXCBWeb). Focused portions of the web are represented as a collection of MWX-

CBWeb compressed bitmaps, each capturing the referencing web pages or those visited web pages. Due to the sparsity of the web, we compress the bitmaps, and use them to mine influential patterns (e.g., frequently referenced web pages, frequently visited web pages) and recommend these web pages to users. Evaluation on our MWXCBWeb model shows its correctness and functionality (in particular, ability to compress bitmaps, interpretability, and flexibility). When applied to real-life web data, our FXCBWeb outperforms many related works in terms of compression ratio, memory consumption, and runtime for interpretable mining of influential patterns from sparse web.

As *ongoing and future work*, we explore further compression and enhancement. We also would apply our solution in mining influential patterns from different sparse webs in the connected world of webs (e.g., web of agents, data, people, things, and trust).

Acknowledgements

This work is partially supported by the (a) Natural Sciences and Engineering Research Council of Canada (NSERC), as well as the (b) University of Manitoba. This work is partially supported by NSERC (Canada) and University of Manitoba.

References

[1] Q. Kong, et al., "Factor space: a new idea for artificial intelligence based on causal reasoning," IEEE/WIC/ACM WI-IAT 2020, pp. 592-599.

[2] R.K. MacKinnon, C.K. Leung, "Stock price prediction in undirected graphs using a structural support vector machine," IEEE/WIC/ACM WI-IAT 2015, vol. 1, pp. 548-555

[3] S.P. Singh, et al., "Analytics of similar-sounding names from the web with phonetic based clustering," IEEE/WIC/ACM WI-IAT 2020, pp. 580-585.

[4] L.V.S. Lakshmanan, et al., "The segment support map: scalable mining of frequent itemsets," ACM SIGKDD Explorations 2(2), 2000, pp. 21-27.

[5] C.K. Leung, et al., "Fast algorithms for frequent itemset mining from uncertain data," IEEE ICDM 2014, pp. 893-898.

[6] C.K. Leung, Y. Hayduk, "Mining frequent patterns from uncertain data with MapReduce for big data analytics," DASFAA 2013, Part I, pp. 440-455.

[7] C.K. Leung, S.K. Tanbeer, "Fast tree-based mining of frequent itemsets from uncertain data," DASFAA 2012, Part I, pp. 272-287.

[8] H. Zheng, P. Li, "Optimizing multi-objective functions in fuzzy association rule mining," IEEE/WIC/ACM WI-IAT 2020, pp. 606-610.

[9] F. Jiang, C.K. Leung, "A data analytic algorithm for managing, querying, and processing uncertain big data in cloud environments," Algorithms 8(4), 2015, pp. 1175-1194.

[10] C.K. Leung, F. Jiang, "Big data analytics of social networks for the discovery of "following" patterns," DaWaK 2015, pp. 123-135.

[11] P. Braun, et al., "Game data mining: clustering and visualization of online game data in cyber-physical worlds," Procedia Computer Science 112, pp. 2259-2268.

[12] P.M.J. Dubois, et al., "An interactive circular visual analytic tool for visualization of web data," IEEE/WIC/ACM WI 2016, pp. 709-712.

[13] C.K. Leung, C.L. Carmichael, "FpVAT: A visual analytic tool for supporting frequent pattern mining," ACM SIGKDD Explorations 11(2), 2009, pp. 39-48.

[14] C.K. Leung, et al., "Visual analytics of social networks: mining and visualizing co-authorship networks," HCII-FAC 2011, pp. 335-345.

[15] C.K. Leung, F. Jiang, "RadialViz: An orientation-free frequent pattern visualizer," PAKDD 2012, Part II, pp. 322-334.

[16] S. Chen, "Internet and beyond: towards a better connected world," IEEE Internet Comput. 24(1), 2020, pp. 36-38.

[17] K. Kundu, A. Dutta, "Multi-agent based distributed mis selection for dynamic job scheduling," IEEE/WIC/ACM WI-IAT 2020, pp. 234-241.

[18] C.K. Leung, et al., "Constrained big data mining in an edge computing environment," in Big Data Applications and Services 2017, pp. 61-68.

[19] A. Hogan, The Web of Data, 2020

[20] C.K. Leung, et al., "Interactive discovery of influential friends from social networks," Social Network Analysis and Mining 4(1), 2014, pp. 154:1-154:13.

[21] C.K. Leung, "Mathematical model for propagation of influence in a social network," Encyclopedia of Social Network Analysis and Mining, 2e, 2018, pp. 1261-1269.

[22] C.K. Leung, et al., "Parallel social network mining for interesting 'following' patterns," CCPE 28(15), 2016, pp. 3994-4012.

[23] Q.M. Rahman, et al., "A sliding window-based algorithm for detecting leaders from social network action streams," IEEE/WIC/ACM WI-IAT 2015, vol. 1, pp. 133-136

[24] K.C. Chatzidimitriou, et al., "Cenote: a big data management and analytics infrastructure for the web of things," IEEE/WIC/ACM WI 2019, pp. 282-285.

[25] A. Kobusinska, et al., "Emerging trends, issues and challenges in Internet of Things, big data and cloud computing," FGCS 87, 2018, pp. 416-419.

[26] R. Sardar, T. Anees, "Web of things: security challenges and mechanisms," IEEE Access 9, 2021, pp. 31695-31711.

[27] F. Buccafurri, et al., "Enabling propagation in web of trust by Ethereum," IDEAS 2019, pp. 9:1-9:6.

[28] C.K. Leung, et al., "Mining 'following' patterns from big sparse social networks," IEEE/ACM ASONAM 2016, pp. 923-930.

[29] C.K. Leung, F. Jiang, "Efficient mining of 'following' patterns from very big but sparse social networks,"

IEEE/ACM ASONAM 2017, pp. 1025-1032.

[30] C.K. Leung, et al., "Mining 'following' patterns from big but sparsely distributed social network data," IEEE/ACM ASONAM 2018, pp. 916-919.

[31] C.K. Leung, et al., "Compression for very sparse big social data," IEEE/ACM ASONAM 2020, pp. 659-666.

[32] C.C.J. Hryhoruk, C.K. Leung, "Compressing and mining social network data," IEEE/ACM ASONAM 2021, pp. 545-552.

[33] K. Wu et al., "Optimizing bitmap indices with efficient compression," ACM TODS 31(1), 2006, pp. 1-38.