קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
מרצה: Dana Shapira

הרצאה 1 – מבוא

הערה: חלק מסיכום הרצאה 1 מבוסס על סיכום של אורנית כהן – oranit95
ציונים: 4 מטלות – 10% מהציון, שאר הבחינה.

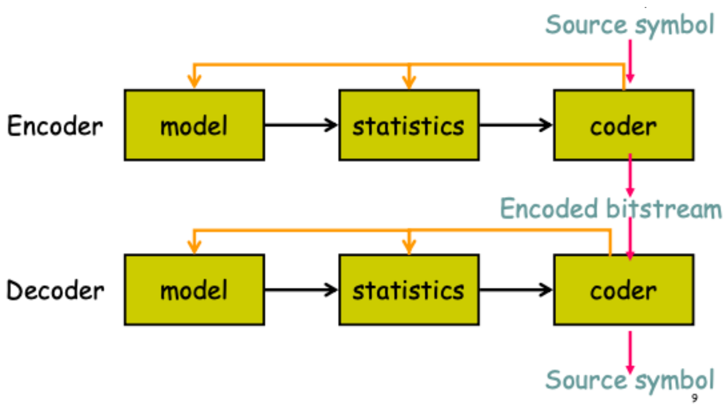
דרישה: הזיכרון לא יקר כמו שהוא היה לפני הרבה שנים אבל בכל אופן יש הגבלות על רוחב הפס, אז או שנרחיב את רוחב הפס שזה לא תמיד אפשרי או להעביר יותר נתונים על אותו רוחב פס. ולכן יש צורך באלגוריתמי דחיסה.

מטרה: שיפור ביצועים גם מבחינת זמנים וגם מבחינת שטח אחסון.

כל מערכת דחיסה מורכבת מ-3 פעולות:
1. שלב המידול – הנחות שעושים על המידע שאיתו אנחנו מתמודדים או שאוספים את המידע על הקובץ. על מנת שהקידוד והדחיסה יהיו מסונכרנים, הקידוד צריך להיות מוכר גם למקודד וגם למפענח.
2. איסוף סטטיסטיקה
3. הקידוד עצמו - רוב הקורס. צריך לדעת את קובץ המקור, מאילו אלמנטים מורכב הקובץ, כלומר:
   - א״ב המקור
   - א״ב ערוצי

לדוגמא: Unary Code: 0,10,110 וכו׳, כאשר בין כל מילת קוד יש שובר אשר הוא ״0״.

כך זה נראה:
יש מקודד (Encoder) ומפענח (Decoder) , בכל שלב אפשר לעדכן את המודל. העדכון צריך להיות מסונכרן עם מה שהמפענח עושה. לוקחים א"ב מהמקור (Source symbol), יוצרים את מילת הקוד והמפענח עושה את הפעולה ההפוכה וממיר את זה חזרה לקוד הרגיל (Source symbol).



טרמינולוגיה (המילים שבשימוש)
- **א״ב המקור** $S := [s_1, s_2, \ldots, s_n]$
- **הסתברות** $P = [p_1, p_2, \ldots p_n]$ כך ש- $\sum_{i=1}^{n} p_i = 1$
  ניתן להניח כי $p_1 \geq p_2 \geq \cdots \geq p_n$
- **מילות קוד** $C = [c_1, c_2, \ldots, c_n]$
  ככל שההסתברות גבוהה יותר כך מילת הקוד קצרות יותר
- **מילות הקוד עולות** $|C| = [|c_1|, \ldots, |c_n|]$
- **אורך מילת הקוד הממוצעת (תוחלת) :** $E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i|$

לדוגמא:

Example $E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i|$

s_i	p_i	Code 1	Code 2
a	0.67	000	00
b	0.11	001	01
c	0.07	010	100
d	0.06	011	101
e	0.05	100	110
f	0.04	101	111
Expected length		3.0	2.22

- Code 1: $|C|=[3,3,\ldots,3]$

יש 2 סוגי דחיסות:

- דחיסות שמאבדות מידע (**lossy** compression)
  אלגוריתמים של דחיסה אשר מאבדים חלקים מהמידע שהיה לפני הדחיסה.
  מיושם בד״כ על קבצי תמונות, ווידאו וקול.
- דחיסות שאינן מאבדות מידע (**lossless compression**)
  אלגוריתמים של דחיסה אשר מאפשרים לפענח את הדחיסה ולקבל **במדוייק** את הקובץ
  לפני הדחיסה, מיושם בדרך כלל על קבצי טקסט.

הערה: הדחיסה ופריסה צריכות להיות פונקציות הפוכות.

קוד חסר רישות – Prefix-free codewords

אף מילת קוד אינה רישא של מילת קוד אחרת, נאמר על קוד אשר מקיימת תכונה זאת ככקוד פרפיקסי.
קוד כזה מאשר מעבר ייחודי (UD) משמאל לימין.

לדוגמא:

$$\epsilon$$
$$0$$
$$01$$
$$011$$
$$0111$$

ניתן לייצג קוד זה ע״י עץ בינארי, כל צלע מיוצגת ב׳0׳ או ב׳1׳, כל עלה בעץ הינו תו כלשהו, כאשר המסלול בין שורש העץ לעלה מייצג את אותו התו, אורך המסלול הוא המסלול מהעץ לעלה.

יתרונות לקוד חסר רישות:

1. קל לקידוד ופענוח
2. ניתן לפיענוח בצורה יחודית UD
3. ניתן להוכיח כי כל דחיסת קוד אופטימלית אשר ניתן ע״י קוד לא חסר רישות אזי ניתן תמיד לדחוס בצורה זהה ע״י קוד חסר רישות ולכן ניתן להתמקד בקוד חסר רישות

קוד Uniquely Decipherable: UD

ניתן לפענוח בצורה יחידה, אם קוד ניתן לפענוח בכמה צורות, קוד זה לא מעניין אם כי לכל קלט יכולים להיות כמה פלטים.

אבחנה:

$$\boldsymbol{Prefix-free \Rightarrow UD}$$

כלומר כל קוד חסר רישות הוא UD

לדוגמא:

עבור מילות הקוד $a=1$, $b=01$, $c=001$, $d=0001$, $e=00001$ נקבל את הקוד הבא: $1|01|001|0001|00001$

$$\boldsymbol{UD \not\Rightarrow Prefix-free}$$

דוגמא: בהינתן המחרוזת $abcde$

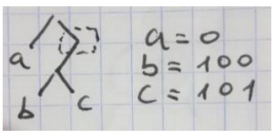
עבור מילות הקוד: $a=1$, $b=10$, $c=100$, $d=1000$, $e=10000$, קוד לא חסר רישות אבל UD: $1|10|100|1000|10000$

כדי להוכיח שקוד כלשהו הוא לא UD יש צורך לתת מחרוזת בינארית שיש לה שתי פירושים שונים

לדוגמא: עבור הקוד: $a=0$, $b=101$, $c=100$, $d=111$, $e=110$, $f=1100$ אזי הקוד

$$1100=1100="f"$$
$$1100=110+0="ea"$$

הערה: עבור דחיסה, קוד **אופטימלי** חייב להיות מיוצג ע״י עץ מלא (לכל צומת יש או 0 בנים או 2 בנים)
הערה: לא כל קוד חסר רישות הוא עץ מלא אבל קוד חסר רישות אופטימלי הוא עץ מלא, לדוגמא:



קוד שלם Complete Code

קוד עבורו כל מחרוזת אינסופית למחצה, ניתנת לפענוח בצורה ייחודית.
איך נדע שקוד הוא לא שלם? מספיק להראות מחרוזת **שאינה** ניתנת לפענוח עם הקוד הזה.

קוד מיידי Instantaneous code

המפענח יודע את הפענוח ברגע שמילת הקוד מסתיימת (משמאל לימין).
זה קורה בקוד חסר רישות (פרפיקסי).
אולם קוד מיידי **אינו** דרוש עבור קוד UD, לדוגמא:
עבור המילה 0111111111111111 עם מילות הקוד $\{0,01,11\}$

מבחן זיהוי יחודי Unique Decipherability Test

הגדרה: יהיו a,b שתי מחרוזות בינאריות כאשר $|a| = k$ ביטים ו- $|b| = n$ כאשר $k < n$
אם k הביטים הראשונים של a הינם זהים ל-k הביטים הראשונים של b אזי הם נקראים **prefix**
ושאר הביטים נקראים **dangling suffix**
לדוגמא: $a = 010, \; b = 01011 \Rightarrow dangling\ suffix = 11$
אלגוריתם:

- Examine all pairs of codewords:
1. Construct a list of all codewords.
2. If there exist a codeword, a, which is a prefix of another codeword, b, add the dangling suffix to the list (if it is not there already), until:
   - I. You get a dangling suffix that is an **original** codeword → the code is not UD
   - II. There are no more unique dangling suffixes → the code is UD

אלגוריתם Sardinas–Patterson

- For given strings S and T, the left quotient is the residual obtained from S be removing some prefix in T.
- Formally $S^{-1}T=\{d \mid ad \in T, a \in S\}$

```
i ← 1
S1 ← C⁻¹C −{ε}
while true
      Si+1 ← C⁻¹Si ∪ Si⁻¹C
      i=i+1
      if ε ∈ Si or c ∈ Si for c in C
            print not UD and exit
      else if ∃ j<i such that Si=Sj
            print UD and exit
```

17

דוגמת הרצה: (קוד UD)
יהיו מילות הקוד $\{0,01,11\}$
1. 0 היא רישא של 01, ולכן dangling suffix = 1
   נעדכן את הרשימה - $\{0,01,11,1\}$
2. 1 היא רישא של 11 ולכן נוסיף את dangling suffix = 1 לרשימה, אולם היא גם ככה ברשימה ולכן אין שינוי.
3. אין עוד מילות קוד שהם רישא של מילת קוד אחרת, ולכן אין יותר dangling suffixes ולכן הקוד הוא UD

הערה: אם היה מילת קוד 1 אז הקוד לא היה UD כי dangling suffix שווה למילת קוד אחרת.

דוגמא להרצה לקוד שהוא אינו ***UD*:**

- Codewords {0,01,10}
- 0 is a prefix of 01 → dangling suffix is 1
- List - {0,01,10,1}
- 1 is a prefix of 10 → dangling suffix is 0 - which is an original codeword!
- → the code is not UD

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ

מרצה: Dana Shapira

קודי יתרות מינמליים *Minimum Redundancy Codes*

קוד הכי יעיל שקיים, כלומר עבור הסתברות מסויימת לא קיים קידוד מעל 0,1 כך שמילת הקוד הממוצעת תיהי קטנה ממנו.

<u>באופן פורמלי:</u>

יהי $E(C,P)$ אורך קוד ממוצע עבור C, אזי C הוא קוד בעל יתירות מינימלי עבור ההסתברות P אם $E(C,P) \leq E(C',P)$ עבור כל קידוד C'

לדוגמא:

s_i	p_i	Code 3
a	0.67	0
b	0.11	100
c	0.07	101
d	0.06	110
e	0.05	1110
f	0.04	1111
Expected length		1.75

Can do even better with Arithmetic coding - 1.65 bits per symbol

המטרה היא לבנות קוד בעל יתירות קוד מינמלית, משמע אורך מילת הקוד הממוצעת היא הכי קטנה שאפשר.

Theorem: Shannon 1948

נגדיר $I(s_i) = -\log_2 p_i$ כאשר $I(s_i)$ זוהי רמת האינפורמציה

לדוגמא עבור הקוד הקודם נקבל כי:

Code 1: $p(s_1)=0.67$, $I(s_1)=0.58$, $p(s_6)=0.04$, $I(s_6)=4.64$

אבחנה: אם $p_i = 1$ אזי $I(s_i) = 0$

אבחנה: ככל שההסתברות גדולה יותר, אזי רמת האינפורמציה קטנה יותר.

נאמר ששני א"ב (אותיות) **בלתי תלויים** אם $I(s_i s_j) = I(s_i) + I(s_j)$

הבעיה כרגע היא איך ניתן להקצות 0.58 ביטים ל-s_1?

לכן *Shannon* הגדיר $H(P) = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$ אשר זהו ממוצע משוקלל של האינפורמציה אשר מהווה **חסם תחתון**, כלומר עבור כל קידוד C נקבל כי $H(P) \leq E(C,P)$

זה נקרא ***Entropy***

נשים לב כי ה-Entropy של הדוגמא הינה 1.65 אשר מהווה חסם תחתון עבור כל קידוד אפשרי.

$-0.67*\log_2 0.67-...-0.04*\log_2 0.04=1.65$

<u>אי-שיוון קראפט Kraft's Inequality</u>

אי-שיוון קראפט מתאר תנאי מספיק והכרחי לשיוך קבוצת מילים לצמתי עץ, כך שלא תשויך יותר ממילה אחת לאורך כל מסלול היוצא מהראש.

שאלה: כמה קצר יכול להיות קוד שהוא UD?

נניח שעבור כל תו s_i יש הסתברות $p_i = 2^{-k_i}$

אזי $I(S_i) = k_i$ אזי לקבוע כל מילת קוד להיות מחרוזת $|c_i| = k_i$ ביטים תגורר למילת הקוד הממוצעת (התוחלת) להיות החסם של ***Shannon***

משפט: יהי $C = [c_1, c_2, \dots, c_n]$ להיות קוד עם n מילות קוד עם אורכים $|C| = [|c_1|, \dots, |c_n|]$

אזי אם C הוא UD אזי

$$K(C) = \sum_{i=1}^{n} 2^{-|c_i|} \leq 1$$

משפט: אם $K(C)=\sum_{i=1}^{n}2^{-|c_i|}\leq 1$ עבור קוד C כלשהו אזי קיים קוד C' אחר כך ש:

1. $E(C,P)=E(C',P)$
2. $|C'|=|C|$
3. C' הוא קוד חסר רישות

משפט: אם $K(C)=\sum_{i=1}^{n}2^{-|c_i|}>1$ עבור קוד C כלשהו אזי הוא **לא קוד חסר רישות**.

e.g. there is no prefix code C with 5 codewords
that satisfy |C|=[2,2,2,2,2]

מטרה: עבור קבוצת הסתברויות נתונה $\mathrm{P}=[p_1,p_2,\dots p_n]$ נרצה לבנות מילות קוד $C=[c_1,c_2,\dots,c_n]$ כך ש:

1. $K(C)\leq 1$
2. $E(C,P)$ **מינמלי**