

אחזור מידע

Ranked retrieval

עד כה, השאלות שלנו היו בוליאניות, כלומר, או שהמסמכים הכילו את התשובה לשאלתנו או שלא. שיטת עבודה זו מתאימה למשתמשים "מומחים", בעלי הבנה מדויקת של צרכיהם ושל אוסף המסמכים, וכן עבור אפליקציות, שיכולות לעבד באופן עצמאי את התוצאות המתקבלות. אולם, עבור רוב המשתמשים צורת עבודה זו לא מתאימה משתי סיבות עיקריות: (1) רוב המשתמשים אינם בקיאים ו/או יכולים לכתוב שאלות בוליאניות (או שאלות מדויקות), ו-(2) לרוב, כמות התשובות החוזרות משאלתא בוליאנית הינה גדולה מאד, ורוב המשתמשים אינם יכולים להתמודד כמויות גדולות של מסמכים.

ליתר דיוק, ברוב המקרים, שאלתא בוליאנית מחזירה או מעט מידי תוצאות (0) או מספר רב של תוצאות (1000). לדוגמא:

□ שאלתא 1: "standard user dlink 650" → 200,000 hits

□ שאלתא 2: "standard user dlink 650 no card found" → 0 hits

לכן, דרושה מיומנות גבוהה בכדי לנסח שאלתא שתחזיר מס' תוצאות שאפשר להתמודד איתן.

במקרה של ranked retrieval, במקום להחזיר קבוצת מסמכים העונה לתוצאות השאלתא, המערכת מחזירה קבוצת מסמכים, כשכל מסמך מקבל "ציון". לרוב, כאשר אנו מתייחסים ל-ranked retrieval, אנו גם מתכוונים לכך שהשאלות שלנו כתובות כטקסט חופשי. כלומר, במקום להשתמש בשפה מיוחדת לכתבת השאלתא (בדומה לשימוש ב- And, or ו-not בשאלות בוליאניות), השאלתא נכתבת כמשפט טבעי בשפה.

כאמור, שאלתא בוליאנית לרוב מחזירה מעט מידי תוצאות (0) או מספר רב של תוצאות (1000). גם במקרה של ranked retrieval המערכת עלולה להחזיר קבוצת מסמכים גדולה (מאד), שרובם לא יהיו בשימוש. אולם כעת יש באפשרותנו להציג למשתמש רק את $k \approx 10$ התוצאות הטובות ביותר, במקום "להפציץ" אותו בתוצאות שכנראה פחות רלוונטיות עבורו.

בכדי שנוכל להחזיר למשתמש, בסדר עולה, מסמכים שהם רלוונטיים ושימושיים למשתמש, אנו צריכים לדרג/למיין את המסמכים שלנו ביחס לשאלתא שלנו. בכדי שנוכל לעשות זאת לכל מסמך אנו ניתן ציון, נניח בין 0 ל-1, המבטא עד כמה המסמך עונה בצורה טובה לשאלתא שלנו.

נניח שיש לנו שאלתא שמורכבת מ-term אחד בלבד. אנו מעוניינים במנגנון למתן ציון הפועל באופן הבא: אם ה-term לא נמצא במסמך, המסמך מקבל ציון 0. ככל שמספר המופעים של ה-term מסמך גדול יותר, המסמך מקבל ציון גבוה יותר.

נבחן מספר אפשרויות לאופן מתן הציון.

Jaccard coefficient

דיברנו על מדד זה בהרצאה 3. JC הינו שיטת נירמול המציינת כמה איברים (terms) משותפים בין שני מסמכים A ו-B (או מסמך ושאלתא), מבין כלל האיברים, ללא כפילויות, שנמצאים בשני המסמכים.

$$JC(A,B) = \frac{|A \cap B|}{|A \cup B|} \text{ הבא: } \text{JC}(A,B)$$

מהגדרה זו, כאשר משווים מסמך עם עצמו, $JC(A,A) = 1$, ואילו כאשר אין איברים משותפים כלל בין שני המסמכים, כלומר $A \cap B = 0$, אז $JC(A,B) = 0$. כזכור, A ו-B אינם צריכים להיות באותו הגודל, והמדד תמיד מחזיר תוצאה בין 0 ל-1.

לדוגמא:

□ Query: *ides of march*

□ $JC=1/6$, Document 1: *caesar died in march*

□ $JC=1/5$, Document 2: *the long march*

הבעייתיות עם JC, היא שהוא אינו מתייחס לתדירות הופעת ה-term במסמך. ישנם terms שתדירות ההופעה שלהם נמוכה (נדירים), ולכן term כזה אם הוא מופיע בשאילתא ובמסמך יש לו יותר משמעות מאשר term שכית. JC אינו מתייחס לנושא זה, ולכן, אנו צריכים מדד טוב יותר, שישקף את הנקודות הנ"ל.

במטריצת מפגשים אנו השתמשנו בערכים בוליאניים בכדי לציין אם term מסוים מופיע במסמך או לא. אנו יכולים להרחיב או לשנות את אופן הייצוג של המטריצה כך, שבמקום להשתמש בערך בוליאני, שמציין אם term קיים או לא, נרשום את מספר המופעים של אותו term במסמך (למטריצה זו אנו קוראים count matrix).

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

באופן זה, אנו יכולים לדעת כמה פעמים מופיע כל term במסמך, ולהתייחס לכך בהתאם. אולם, השימוש בוקטורים אינו משמר את מיקום ה-terms במסמך. לדוגמא, המסמכים John is quicker than Mary ו-Mary is quicker than John הם בעלי אותו הווקטור. אנו קוראים למצב הזה bag of words model. יש לנו מסמכים שהם בעלי אותו הווקטור, כלומר, עבוד שאילתא מסוימת, אם מסמך אחד עונה עליה, אז גם השני עונה עליה באותה המידה. נראה, אם כך, שחזרנו שלב אחד אחורנית. היה לנו פתרון לבעיה זו והוא השימוש ב-positional index, אפשר לנו להבדיל בין שני המסמכים. אנו נראה פתרון לכך בהמשך.

Term frequency tf

ה-term frequency, $tf_{t,d}$, של term t במסמך d, מוגדר כמספר הפעמים ש-t מופיע ב-d. מסמך שבו term מסוים מופיע 10 פעמים יותר רלוונטי ממסמך שבו ה-term מופיע פעם אחת בלבד, אבל, הוא לא רלוונטי פי 10. רלוונטיות של מסמך היא לא פרופרציונאלית למס' הפעמים ש-term נתון מופיע בו. בהתאם לכך, אנו מעוניינים להשתמש בממד ה-term frequency בעת חישוב ציון המסמך. כיצד נעשה זאת ?

אנו נגדיר מדד חדש המבוסס על ה-ft (מבוסס על תצפיות אימפריות), מדד זה הוא ה-Log-frequency weighting של term t במסמך d, הוא מוגדר כ:

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & tf_{t,d} > 0 \\ 0 & otherwise \end{cases}$$

בהתאם למדד זה, אם term מופיע 0 פעמים, אז $w_{t,d} = 0$, אם term מופיע פעם אחת, אז $w_{t,d} = 1$, באופן דומה אם הוא מופיע פעמיים אז $w_{t,d} = 1.3$, אם הוא מופיע 10 פעמים אז $w_{t,d} = 2$, אם הוא מופיע 1000 פעמים אז $w_{t,d} = 4$, וכך הלאה.

את הציון הכללי של המסמך נחשב ביחס לשאילתא באופן הבא: נסכום את ה-Log-frequency weighting של כל ה-terms שמופיעים גם בשאילתא וגם במסמך.

$$Score = \sum_{t \in q \cap d} (1 + \log_{10} tf_{t,d})$$

במידה ואף אחד מה-terms שמופיעים בשאילתא לא מופיע במסמך, אנו מקבלים ציון 0.

Document frequency

כאמור, terms שהינם נדירים יש להם יותר משמעות מ-terms שכיחים (למשל stop words). נניח שיש לנו term נדיר בשאילתא. מסמך שמכיל את ה-term הזה הוא, ככל הנראה, מסמך רלוונטי עבורנו. במקרים כגון אנו, אנו רוצים שה-terms הנדירים הללו יהיו בעלי משקל גבוה יותר מ-terms שכיחים. כמו כן, נניח שיש לנו שאילתא שמכילה

terms שהינם שכיחים. מסמכים שמכילים את ה-terms הללו הינם יותר רלוונטיים ממסמכים שאינם מכילים את ה-terms כלל. אנו רוצים שגם terms כאלו יהיו בעלי משקל גבוהה יחסית, אך נמוך מאשר של terms נדירים.

נגדיר מדד חדש, df_t , שהוא document frequency של t , כלומר מספר המסמכים המכילים את t (והוא זהה לאורכו של ה-posting list של אותו ה-term כאשר אנו משתמשים ב-non-positional index). ערכו של df_t אינו גבוה ממספר המסמכים שיש לנו, $df_t \leq N$. למעשה, ככל ש-term יותר נדיר, כך הוא נמצא בפחות מסמכים, ובהתאם ערכו של ה- df_t נמוך יותר. בהתאם לכך, אנו מגדירים את idf (inverse document frequency) של t באופן הבא:

$$idf_t = \log_{10} \left(\frac{N}{df_t} \right)$$

כאשר אנו משתמשים ב-log בכדי לדכא את האפקט של idf . באופן זה, ככל שה-term t נמצא בפחות מסמכים, כך ערכו של idf_t עולה, כפי שרואים בדוגמא הבאה (שבה $N=1,000,000$).

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

האם ל- idf יש השפעה על הציון שמקבל מסמך כשהשאלתא בנוייה מ-term אחד? לא, ל- idf אין השפעה על הציון שמקבל מסמך כאשר השאלתא מורכבת מ-term אחד. idf משפיע על ציון המסמך כאשר השאלתא מורכבת מלפחות שני terms, מכיון שהמטרה של מדד זה היא להביא לידי ביטוי את השכיחות של כל אחד מה-terms של השאלתא במסמך, ולכן, אם השאלתא בנוייה מ-term אחד, אז אין למדד זה משמעות. לדוגמא, עבור השאלתא capricious person, idf מתייחס למספר המופעים של capricious בתוצאה הסופית, יותר מאשר מספר המופעים של person.

ה-collection frequency של t הוא מספר המופעים של t באוסף המסמכים, כולל חזרות, בניגוד ל-document frequency של t , כלומר מספר המסמכים המכילים את t . מדוע השתמשנו ב-inverse document frequency ולא ב-inverse collection frequency:

לצורך ההסבר נשתמש בדוגמא הבאה:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

בדוגמא זו, ה- cf של insurance הוא 10440 ואילו ה- cf של try הוא 8760. בהתאם לכך, איזה term יותר טוב בחיפוש, ויקבל ציון גבוה יותר? אם אנו מסתכלים על ה- cf אנו רואים שהערכים של שני ה-terms דיי דומים, ולכן קשה להעריך מי מהם עדיף. אבל אם מסתכלים על ה- df , רואים ש-insurance הינו נדיר יותר מ-try, ולכן הוא עדיף בחיפוש, ויקבל ציון גבוה יותר.

tf.idf weighting

ה- $tf.idf$ של term שווה למכפלה של ה- tf שלו ב- idf שלו. זהו מדד המשלב את מספר המופעים של term מסוים באוסף המסמכים ואת הנדירות של אותו ה-term.

$$w_{t,d} = \log(1 + tf_{t,d}) \times \log_{10} \left(\frac{N}{df_t} \right)$$

מדד זה הוא המדד הטוב ביותר בתחום אחזור המידע על מנת לתת ציונים למסמכים בהתאם לשאלות. ערכו של מדד זה עולה כלל שמספר ההופעות של term גדל במסמך, כמו כן, הוא גדל ככה שה-term נדיר יותר באוסף המסמכים שלנו.

בהתאם לכך, ציון של מסמך בהינתן שאלתא מוגדר כ:

$$Score(q, d) = \sum_{t \in q \cap d} tf \cdot idf_{t,d}$$

שקיימות הרבה גרסאות לחישוב זה. למשל, איך אנו מחשבים את tf (עם או בלי log), האם ל-terms בשאלתא יש משקלים, וכו'.

אנו יכולים לחזור למטריצת מפגשים, כאשר הפעם נרשום את ערך ה- $tf \cdot idf$ של כל term במסמך.

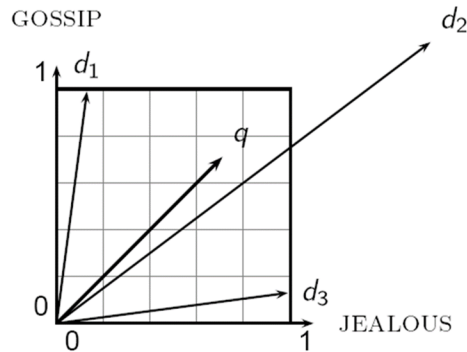
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

מסמכים כווקטורים

אנו יכולים להתייחס לאוסף המסמכים שלנו כמרחב ווקטורי בעל $|V|$ מיימדים, כאשר ה-terms הם צירים באותו המרחב ואילו המסמכים הם נקודות או ווקטורים במרחב. כאשר מדובר על חיפוש ב-web, אז המרחב הווקטורי בעל מס' מיימדים מאד גדול. כמו כן, הווקטורים הללו מאד דלילים.

בכדי לבצע שאלות, אנו תחילה נייצג שאלות כווקטורים במרחב (אפשר להתייחס לשאלתא עצמה כאל מסמך מאד קטן), ואח"כ, נדרג כל מסמך בהתאם לקרבתו לווקטור השאלתא במרחב. כאשר קירבה, המרחק ההופכי, מייצגת את הדמיון בין הווקטורים. באופן זה אנו יכולים להימנע משאלות בוליאניות, וכן אנו נותנים ציון גבוה יותר למסמכים שיותר רלוונטיים לנו, וציון נמוך למסמכים שלא רלוונטיים לנו.

בהתאם לכך, כיצד נגדיר מרחק? המרחק האוקלידי, כפי שניתן לראות מהדוגמא, בין \vec{q} ובין \vec{d}_2 הוא גדול, למרות שהתפלגות ה-terms בשאלתא \vec{q} והתפלגות ה-terms ב- \vec{d}_2 מאד דומות.



נבצע ניסוי תאורטי. ניקח מסמך d ונוסיף אותו לעצמו. נקרא למסמך החדש d' . "סמנטית", ל- d ול- d' יש את אותו התוכן, אולם המרחק האוקלידי בין שני המסמכים הוא יחסית גדול. אם נשתמש בייצוג פולארי, אנו נראה שהווקטור d' גדול פי 2 מהווקטור d , אולם הזווית בניהם היא 0, והיא מתאימה לזהות מקסימאלית בין המסמכים. אם כך,

נדרג את המסמכים בהתאם לזווית שבניהם. ומכאן, שבכדי לדרג את המסמכים בסדר יורד של הזווית בין השאילתא והמסמך אנו יכולים לדרג את המסמכים בסדר עולה של ערך הקוסינוס של הזווית בין וקטור השאילתא ווקטור המסמך.

מתמטית, ניתן לנרמל את אורכו של וקטור ע"י חלוקת כל אחד ממרכיביו באורכו. לשם כך נגדיר את L_2 באופן הבא:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

אם נחלק את הווקטור ב- L_2 שלו, נקבל את ווקטור היחידה. אם נסתכל על d ו- d' , נראה שיש להם את אותו ווקטור לאחר הנרמול.

cosine(query,document)

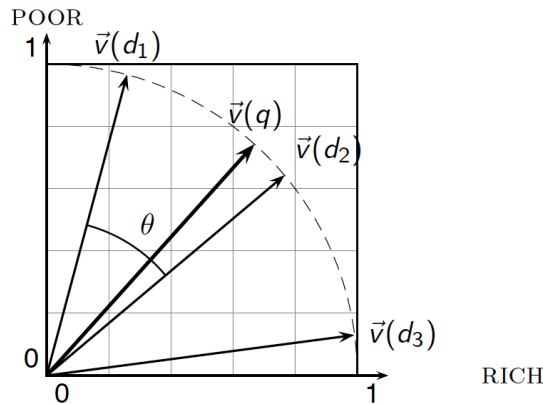
את ערך הקוסינוס בין השאילתא q והמסמך d אנו מחשבים באופן הבא:

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| |\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \cdot \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|\mathcal{V}|} q_i d_i}{\sqrt{\sum_{i=1}^{|\mathcal{V}|} q_i^2} \sqrt{\sum_{i=1}^{|\mathcal{V}|} d_i^2}}$$

וזאת כאשר q_i הוא tf.idf של i term בשאילתא ו- d_i הוא td.idf של i term במסמך.

עבור הווקטור המנורמל, הערך קוסינוס הדימיון הוא ה-dot product (תזכורת אם A ו- B הם ווקטורים אז $A \cdot B = A_1 B_1 + A_2 B_2 + \dots + A_n B_n$), כמו כן, אם $\|A\|$ מציין את אורכו של הווקטור A , ו- $\|B\|$ מציין את אורכו של הווקטור B , אז $A \cdot B = \|A\| \|B\| \cos(\theta)$, כאשר θ היא הזווית בין A ל- B .

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_{i=1}^{|\mathcal{V}|} q_i d_i$$



דוגמא:

עד כמה דומים המסמכים? (לצורך הפשטות אנו לא נחשב את ה-idf).

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

יש לנו שלושה ספרים, SaS: *Sense and Sensibility*, PaP: *Pride and Prejudice*, ו-WH: *Wuthering Heights*.

לצורך הדוגמה אנו נתייחס לארבעה term-ים בספרים אלו. בטבלה מוצגים ה-term frequency עבור כל אחד מה-term-ים בארבעת הספרים שלנו. אנו צריכים לחשב את ה-tf וכן את ה-idf (בדוגמה אנו נניח שהערכי ה-idf הם 1 עבור כל ה-term-ים), ולהכפיל בין התוצאות. לדוגמה, עבור ה-term affection בספר SaS, ערכו של ה-tf הוא $ft = 115$. הערכים של tf עבור שאר ה-term-ים בספרים השונים מובאים בטבלה Log frequency weighting. בשלב הבא אנו נחשב עבור כל ווקטור (עמודה בטבלה) את ערכו המנורמל, ובהתאם את ערכי ה-term-ים המנורמלים. למשל, עבור הווקטור הראשון, SaS, אורכו הוא $\sqrt{3.06^2 + 2^2 + 1.3^2 + 0^2} = \sqrt{15.0536} = 3.88$, שאר הערכים המנורמלים מובאים בהתאם לכך, ערכו המנורמל של ה-term affection בספר SaS הוא $\frac{3.06}{3.88} = 0.789$. בטבלה After length normalization.

Log frequency weighting

After length normalization

term	SaS	PaP	WH	term	SaS	PaP	WH
affection	3.06	2.76	2.30	affection	0.789	0.832	0.524
jealous	2.00	1.85	2.04	jealous	0.515	0.555	0.465
gossip	1.30	0	1.78	gossip	0.335	0	0.405
wuthering	0	0	2.58	wuthering	0	0	0.588

נשתמש בנוסחה $\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_{i=1}^{|V|} q_i d_i$ בכדי לחשב את הדמיון בין המסמכים השונים.

$$\cos(\text{SaS}, \text{PaP}) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94$$

$$\cos(\text{SaS}, \text{WH}) \approx 0.79$$

$$\cos(\text{PaP}, \text{WH}) \approx 0.69$$

בהתאם לתוצאות, SaS ו-PaP הוא הספרים הדומים ביותר (או הקרובים ביותר) אחד לשני.

לסיכום, אופן ביצוע השאילתא מתואר באלגוריתם הבא:

COSINESCORE(q)

```

1 float Scores[N] = 0
2 float Length[N]
3 for each query term t
4 do calculate  $w_{t,q}$  and fetch postings list for t
5   for each pair( $d, tf_{t,d}$ ) in postings list
6     do Scores[d] +=  $w_{t,d} \times w_{t,q}$ 
7 Read the array Length
8 for each d
9 do Scores[d] = Scores[d]/Length[d]
10 return Top K components of Scores[]

```

הטבלה הבאה מסכמת מס' אפשרויות לחישוב ה-Term frequency, ה-Document frequency וביצוע הנירמול.

Term frequency		Document frequency		Normalization	
n (natural)	$tf_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(tf_{t,d})$	t (idf)	$\log \frac{N}{df_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-df_t}{df_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^\alpha$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$				

הרבה מנועי חיפוש מאפשרים שימוש של משקלים שונים בין השאילתא למסמכים. SMART – מייצג את הקומבינציה שבשימוש ע"י מנוע החיפוש, כשהסימול הוא בפורמט ddd.qqq (כל שלשה היא בהתאם לאותיות המופיעות בטבלה למעלה). לרוב נעשה שימוש ב-inc.ltc, שמשמעו, עבור המסמך logarithmic term frequency, no document frequency ו-cosine normalization. ועבור השאילתא logarithmic term frequency, idf document frequency ו-cosine normalization.

לדוגמא:

המסמך: car insurance auto insurance

השאילתא: best car insurance

Term	Query						Document				Pro d
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

$$\text{Doc length} = \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$$

$$\text{Score} = 0+0+0.27+0.53 = 0.8$$