אחזור מידע

Scoring and results assembly

את ההרצאה הקודמת סיימנו עם האלגורית הבא לחישוב ציוני מסמכים המתבסס על ערכי הקוסינוס.

```
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} × 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[]
```

כאשר אנו שואלים שאילתא, אנו למעשה רוצים למצוא את k המסמכים באוסף המסמכים "הקרובים ביותר" לשאילתא שלנו. כלומר את k המסמכים עם ערך ה-cosine הגבוה ביותר.

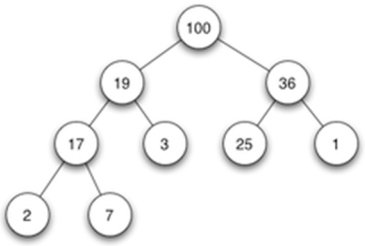
בכדי לבצע פעולה זו באופן יעיל אנו צריכים למצוא דרך לחשב את ערך ה-cosine לכל מסמך בצורה יעילה. ובהתאם, לבחור את k המסמכים עם ערך ה-cosine הגבוה ביותר, גם כן באופן יעיל.

השאלה היא, האם ניתן לעשות זאת מבלי לחשב את כל N ערכי ה-cosine (של כל המסמכים בקורפוס) ? (כן, אנו צריכים להתייחס רק למסמכים שנמצאים ב-posting lists של כל אחד מה-terms שמופיעים בשאילתא. אבל אנו רוצים משהו יעיל יותר מזה).

מה למעשה אנו עושים ? אנו פותרים את בעיית k השכנים הקרובים ביותר עבור ווקטור השאילתא. באופן כללי, אנו לא יודעים איך לפתור בעיה זו באופן יעיל כאשר יש לנו הרבה מסמכים ושאילתא ארוכה. אבל, בעיה זו פתירה עבור שאילתות קצרות (עם מספר מועט של terms), ואינדקסים סטנדרטיים תומכים בה בצורה טובה.

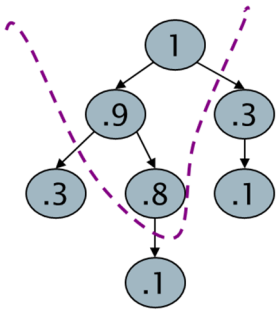
נסתכל על מקרה מיוחד, שהו אין משקלים לביטויים בשאילתא (אנו מניחים שכל ביטוי בשאילתא מופיע פעם אחת בלבד). במקרה כזה, לצורך הדרוג, אנו לא צריכים לנרמל את ווקטור השאילתא. ובהתאם האלגוריתם הוא מעט פשוט יותר מזה שהוצג בהרצאה הקודמת (בשורה 6 של האלגוריתם $w_{t,q}$ הוא 1).

ברור במקרים אנו רוצים לקבל את k המסמכים בעלי הדרוג הגבוה ביותר (לפי cosine ranking), ואנו רוצים לעשות זאת מבלי לחשב את ה-cosine ranking לכל מסמך ומסמך ומבלי למיין את כל המסמכים. נניח ש-J מציין את המסמכים להם ה-cosine ranking שונה מ-0 (מס' זה יכול להיות גבוה מאד), אנו רוצים לקבל את K המסמכים הטובים ביותר מתוך J. בכדי לעשות זאת ביעילות טובה יותר מ-$j \log j$ (היעילות של מיון), אנו יכולים להשתמש במבנה נתונים שנקרא heap. Heap הוא מבנה נתונים בצורת עץ בינארי המקיים תכונה בסיסית: המפתח של כל צומת בעץ קטן ממפתח אביו (בערימת מקסימום). כתוצאה מדרישה זו, הצומת בעל המפתח הגדול ביותר הוא תמיד השורש של העץ, וניתן למצוא אותו בסיבוכיות זמן $O(1)$ (כלומר במספר פעולות קבוע, שאיננו תלוי במספר האיברים בערימה).



בכדי לבנות את העץ אנו צריכים לבצע $2 \times J$ פעולות. ואח"כ, בשביל לבחור את k המסמכים הטובים ביותר $\log(2 \times J)$ פעולות לכל מסמך.

עבור J=1M ו-K=100, התהליך הזה לוקח בערך 10% מהעלות של מיון.



הבעיה העיקרית שלנו או במילים אחרות, צוואר הבקבוק העיקרי שלנו הוא חישוב ה-cosine. זה חישוב שלוקח הרבה זמן. ולכן, האם אנו יכולים להימנע מכל החישובים האלו ? כן, ניתן להשתמש בהיוריסטיקות שונות שיתנו לנו ערכים קרובים לחישוב שלנו (באופן מהיר ופשוט יותר), אבל אז אנו עלולים לקבל תשובות שגויות. במקרים כאלו, מסמך שלא נמצא ברשימת k המסמכים הטובים ביותר יכול להופיע ככזה. אבל, האם זה כל כך נורא ?

למשתמש יש משימה ממנה מגזרת השאילתא. ערך ה-cosine מוחשב למסמכים בהתאם לשאילתא, ולפיכך, ערך ה-cosine מתאר עד כמה המשתמש "שמח" עם התוצאה (המסמך) שקיבל. אבל, אנו מניחים שאם מסמך מכיל את ה-terms שמופיעים בשאילתא (בהתאם לתדירות והנדירות) אז הוא מסמך רלוונטי – אבל יתכן שאנו טועים והוא לא רלוונטי. ולכן, אם נקבל רשימה של k מסמכים "קרובים" ל-k המסמכים הטובים ביותר לפי ה-cosine, אז אנחנו במצב טוב. ואין זה נורא עם יהיו כמה תשובות שגויות ברשימת k המסמכים הטובים ביותר.

אם כך, אופן הפעולה שלנו יהיה באופן הבא: נמצא קבוצה A של מועמדים, כש-$K < |A| << N$ (A לא חייב לכלול את K המסמכים הטובים ביותר, אבל הוא מכיל הרבה מ-K המסמכים הטובים ביותר). וכתשובה נחזיר את K המסמכים הטובים ביותר מ-A. ניתן ליישם גישה זה אם פונקציות דרוג שונות (לא בהכרח cosine). בהמשך נציג מספר סכמות שעובדות בצורה זו.

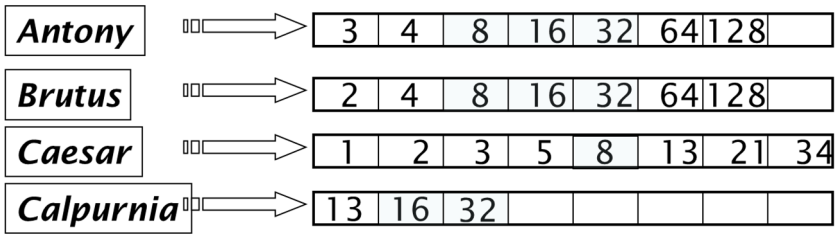
Index elimination

אלגוריתם חישוב ה-cosine הבסיסי מתייחס רק למסמכים שבהם יש לפחות term אחד מהשאילתא. ניתן להרחיב גישה זו. אפשרות אחת היא שנתייחס רק ל-terms בשאילתא עם high-idf. אפשרות נוספת היא שנתייחס רק למסמכים המכילים מספר terms מהשאילתא.

נניח שנתונה השאילתא: catcher in the rye. אינטואיטיבית, in ו-the תורמים מעט לציון (הציון שלהם קרוב ל-0), ולכן הם כמעט ולא משנים את הדרוג. ולכן, אנו צריכים לסכום רק את הציונים המתקבלים מ-catcher ומ-rye.

מכוון שביטויים בעלי low-idf נמצאים בהרבה מסמכים, אם נתעלם מביטויים אלו, המסמכים הללו לא יופיעו בקבוצה A.

כל מסמך המכיל לפחות ביטוי אחד מהביטויים שבשאילתא הוא מועמד להיות חלק מ-K המסמכים הטובים ביותר. אם השאילתא מכילה מספר ביטויים, אנו נחשב את הציון של מסמכים המכילים יותר מביטוי אחד מכלל הביטויים של השאילתא.



בדוגמא, אם נניח שאנו מתייחסים לפחות ל-3 מתוך 4 terms, אנו נחשב את הציון רק עבור מסמכים מס' 8, 16 ו-32. שיטה זו קלה ליישום.

Champion lists

בשיטה זו, לכל term במילון אנו נחשב/נמצא את r (אנו צריכים לבחור את r בזמן בניית האינדקס) המסמכים עם הציון (tf.idf) הגבוה ביותר. למסמכים האלו אנו קוראים Champion lists (או fancy list או top docs). במקרה זה, יתכן ו-r יהיה קטן מ-K. בעת ביצוע השאילתא, נחשב את הציון רק עבור מסמכים שנמצאים ב- Champion lists(מספיק שעבור term מסוים, מסמך כלשהו יהיה ב-champion list בשביל שנתייחס אליו, אין צורך שאותו מסמך יהיה ב-champion list של כל ה-terms), בהתאם לביטויים שבשאילתא. K המסמכים הטובים ביותר יהיו מבין המסמכים ששייכים ל-Champion list .

Static quality scores

עד עכשיו התייחסנו לכל המסמכים כזהים בחשיבות. אבל אין זה כך. נניח שישנם שני מסמכים בנושא מסוים, אחד נכתב ע"י תלמיד בבית ספר, ואילו השני נכתב ע"י מרצה באוניברסיטה. במקרה זה, שני המסכים רלוונטיים בהקשר לנושא, ואולם אנו רוצים להתייחס גם לסמכותיות של המסמך. במקרה זה, המסמך שנכתב ע"י מרצה האוניברסיטה הוא יותר סמכותי. אפשר לסכם זאת כך: אנו רוצים שהמסמכים עם הדרוג הגבוה יהיו גם רוולטים וגם סמכותיים. את הרלוונטיות של המסמך אנו מודדים ע"י ציון ה-cosine. אבל, סמכותיותו של מסמך היא תכונה שאינה תלויה בשאילתא. להלן מס' דוגמאות שיכולות לציין שמסמך הוא מוסמך: דפי ויקיפדיה, מאמר בעיתונים מסוימים, מאמר עם מספר רב של ציטוטים, הרבה הפניות לדף ווב מסוים וכו'.

בכדי שנוכל להתייחס לסמכותיותו של מסמך, לכל מסמך אנו נשייך מדד, g(d) (goodness), שמציין את סמכותיות המסמך כערך בין 0 ל-1, ללא תלות בשאילת. למשל, מספר הציטוטים של מסמך מסויים יתורגם לערך בין 0 ל-1. אנו רוצים לאחד את ציון ה-cosine וציון הסמכותיות לערך אחד. אפשרות אחת היא ע"י סכימת שני הערכים: net-score(q,d) = g(d) + cosine(q,d), או בעזרת כל בקומבינציה לינארי אחרת. בהתאם, אנו נחפש את K המסמכים בעלי ה-net score הגבוה ביותר.

K המסמכים הטובים ביותר – שיטה מהירה

נמיין את כל ה-postings לפי g(d) (ולא לפי ה-docID). אם כל מסמך מקבל ערך g(d) ייחודי (כלומר, אין שני מסמכים עם אותו הערך), אנו יכולים לעבור על רשימות ה-postings של הביטויים בשאילתא ובמקביל לבדוק חיתוכים בין הרשימות השונות או לבצע את חישוב ערך ה-cosine. גם אם הערכים של g(d) הם לא ייחודיים, אנו יכולים לבצע את העבודה במקביל, רק שעכשיו, במידה ואנו נתקלים בשני ערכי g(d) זהים, אנו צריכים גם לבדוק אם יש להם docID זהה. אבל למה לעשות זאת ? מה היתרון בצורת עבודה זו ? אם נבצע את המיון בהתאם ל-

g(d), אז סביר שמסמכים בעלי דרוג גבוה יופיעו בתחילת ה-posting list. במקרים שזמן החיפוש מוגבל, זה מבטיח לנו שאנו עוברים על המסמכים בעלי הדרוג הגבוה ראשונים.

אנו יכולים למזג את ה-champion lists עם ערך ה-g(d). במקרים אלו, לכל ביטוי אנו שומרים champion lists של r המסמכים עם ערך g(d) +tf-idftd הגבוה ביותר. וכן, אנו נחפש את K המסמכים הטובים ביותר רק ברשימת ה-champion list.

High and low lists

לכל term אנו שומרים שתי רשימות (או posting lists) הנקראות High ו-Low. הרשימה הראשונה מכילה מסמכים בעלי חשיבות גבוהה, ואילו הרשימה השנייה מכילה מסמכים בעלי חשיבות נמוכה. ניתן לחשוב על High כ-champion list. כאשר אנו עוברים על ה-posting list במהלך ביצוע השאילתא, אנו נעבור תחילה על רשימת ה-high. אם נקבל מרשימת ה-high K או יותר מסמכים, ניקח את K הטובים ביותר, אחרת, נמשיך עם רשימת ה-low. לבניית הרשימות הללו אפשר להשתמש רק עם ערך ה-cosine, ואין צורך ב-g(d).

Impact-ordered postings

אנו רוצים לחשב את הציון רק עבור מסמכים שיש להם ערך $wf_{t,d}$ גבוה מספיק. אם אנו נמיין את ה-postings בהתאם לערך ה-$wf_{t,d}$, בעת מעבר על ה-postings נוכל לעצור כשנגיע למסמך שערך ה- ה-$wf_{t,d}$ לא גבוה מספיק. אולם יש לנו בעיה בשיטה זו, רשימות ה-postings השונות לא ממוינות באותו הסדר, ולכן לא ניתן לבצע עבודה במקביל. אם כך, איך נחשב את הציונים בכדי לבחור את K המסמכים הטובים ביותר. יש לנו שני רעיונות לשם כך.

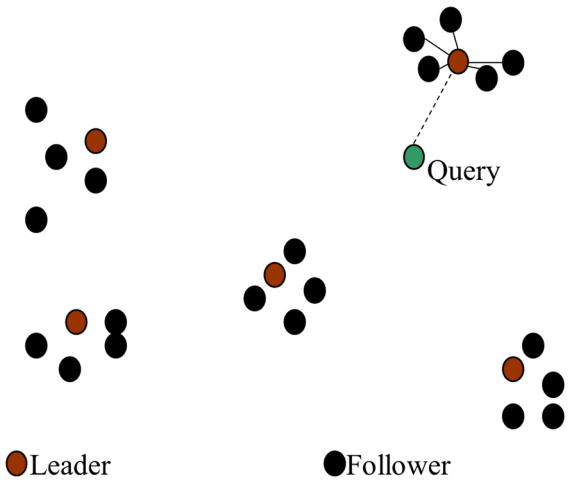
כאשר אנו עוברים על רשימת ה-postings של הביטוי t (שממויין לפי $wf_{t,d}$), אנו נעצור (1) כאשר הגענו למספר r, מוגדר מראש, של מסמכים או (2) ערכו של $wf_{t,d}$ יורד מתחת לסף קבוע מראש. אנו לוקחים את האיחוד של קבוצות המסמכים (קבוצה עבור כל ביטוי) ומחשבים את הציון עבור המסמכים באיחוד הנ"ל.

כאשר אנו עוברים על רשימת ה-postings נעשה זה בהתאם לערך ה-idf, מהגבוה לנמוך, כשערך idf גבוה כנראה תורם הרבה לחישוב הציון (למעשה אנו מתחילים עם הביטויים הנדירים יותר). כשאנו מעדכנים את ערך הציון של מסמך, בהתאם לביטויים השונים של השאילתא, אנו נעצור אם השינוי בערך החדש הוא זניח. במקרה זה, אפשר להשתמש בערך ה-cosine או בערך net score אחר.

Cluster pruning

נבחר $\sqrt{N}$ מסמכים באופן אקראי, ונקרא להם מנהיגים (leaders). לכל מסמך אחר, נחשב מראש מיהו המנהיג הקרוב ביותר אליו (למשל ע"י חישוב cosine). מסמך המשויך למנהיג הוא העוקב שלו. לכל מנהיג יש כ-$\sqrt{N}$ עוקבים.

ביצוע השאילתא יתבצע באופן הבא: (1) עבור שאילתא Q, נמצא את המנהיג L הקרוב ביותר אליה. (2) ניקח את K המסמכים הקרובים ביותר ל-L מבין העוקבים שלו.



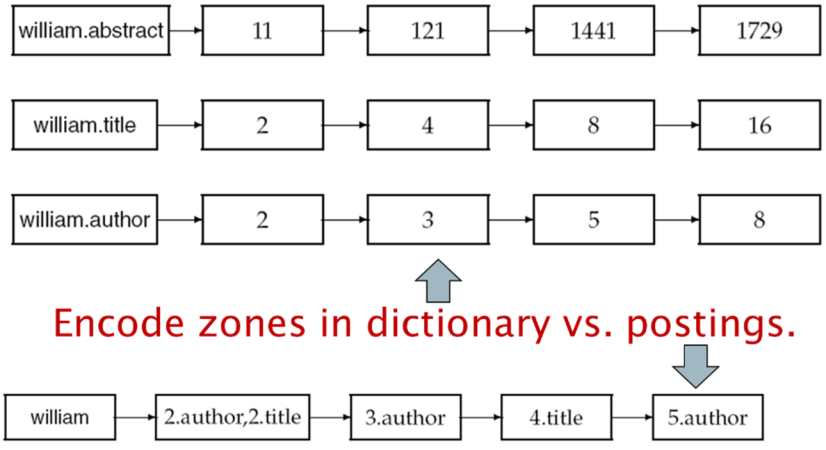
למה אנו בוחרים את המנהיגים באופן אקראי ? יש לכך שתי סיבות: (1) שיטה זו מהירה. (2) באופן זה, המנהיגים משקפים את התפלגות המידע.

אפשר להכליל שיטה זו. לדוגמא, כל מסמך משויך ישויך ל-b1=3 (נניח) מנהיגים קרובים ביותר (כלומר, לכל מסמך אין מנהיג אחד בלבד). באופן דומה, בעת ביצוע שאילתא, עבור השאילתא עצמה, נמצא את b2=4 (נניח) המנהיגים הקרובים ביותר ואת המסמכים המשויכים אליהם. באופן זה, כמות המסמכים שאנו צריכים לבדוק גדולה יותר, אבל ככל הנראה, אנו נגדיל את ה-recall של התוצאה המתקבלת.

<u>אינדקסים פרמטריים ואזוריים</u>

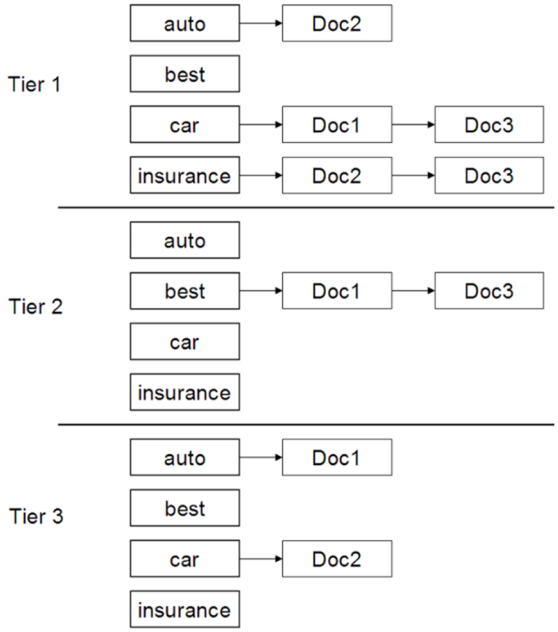
עד כה, התייחסנו למסמך כאוסף של terms. אולם, במציאות, למסמך יש מספר חלקים, להם משמעות סמנטית שונה, כגון שם המחבר, כותרת, תאריך פרסום, שפה, פורמט וכו'. נתונים אלו מרכיבים את המטה-דאטה של המסמך. ישנם מקרים שהם אנו מעוניינים לחפש בהתאם למטה-דאטה. לדוגמא, מצא מסמכים שנכתבו ע"י ויליאם שייקספיר בשנת 1601, המכילים alas poor Yorick. בדוגמא זו, Year=1601 הוא שדה, וכך גם שם משפחת המחבר, שייקספיר, הוא שדה. בכדי לבצע שאילתות על אותו מטה-דאטה, אנו יוצרים אינדקסים על שדות או פרמטרים שבו יש posting עבור כל ערך בשדה.

Zone (אזור) הוא קטע במסמך, כגון כותרת, אבסטרקט או בביליוגרפיה, המכיל כמות כלשהי של טקסט (בניגוד לשדה שבו מס' מילים מועט). אנו נבנה inverted index גם עבור ה-terms הנמצאים ב-zones השונים, על מנת לאפשר חיפוש בהם. לדוגמא, "find docs with merchant in the title zone and matching the query gentle rain"



Tiered indexes

במקום שיהיה לנו posting list אחד, נחלק את ה-postings להיררכיה של רשימות, מהחשובה ביותר עד להכי פחות חשובה. בכדי לבנות את ההיררכיה הזאת ניתן להשתמש ב-g(d) או בכל מדד אחר. את ה-inverted index אנו מחלקים לקבוצות בעלי חשיבות פוחתת. בזמן השאילתא, אנו נשתמש בקבוצה ברמה הגבוהה ביותר, אך אם לא נקבל K מסמכים, נרד לרמה נמוכה יותר.

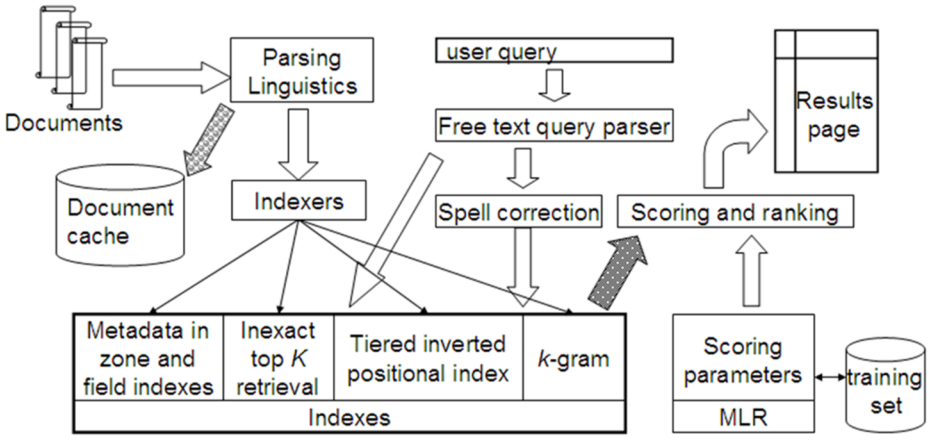


Query Term Proximity and Query Parsing

שאילתות טקסט חופשי הן אוסף של terms המוזנים למנוע החיפוש (נפוץ בחיפוש באינטרנט). המשתמשים מעדיפים מסמכים בהם ה-terms מופעים בקרבה יחסית אחת לשני. נגדיר כ-w את החלון (מס' ה-terms הרצופים) הקטן ביותר במסמך המכיל את כל ביטויי השאילתא. לדוגמא, עבור השאילתא "strained mercy", החלון הקטן ביותר במסמך "The quality of mercy is not strained" הוא 4 (מילים). אנו מעוניינים בפונקציית דרוג שתיקח ערך זה בחשבון.

שאילתות טקסט חופשי יכולות להתפרש כמספר שאילתות. נניח שיש לנו את השאילתא "rising interest rates". בשלב הראשון נבצע את השאילתא כ-phrase query. אם כמות המסמכים המוחזרת קטנה מ-k, נפצל את השאילתא לשתי שאילתות נוספות, לדוגמא, "rising interest" ו-"interest rates". אם עדיין כמות המסמכים המוחזרים קטנה מ-k, נבצע 3 שאילתות נוספות, אחת עבור כל term. נדרג את המסמכים שהתקבלו מהשאילתות לפי vector space scoring.

ראינו שפונקציית הדרוג יכולה לחבר ציונים שונים כגון cosine, static quality, קירבה וכו' (לתת לכל שיטה אחוז כלשהו מהציון הסופי). אולם, כיצד אנו יכולים לדעת מה הצרוף הטוב ביותר ? ובכן, בחלק מהמקרים, המערכת מכוונת ע"י אנשים (כלומר, אנשים עוברים על המערכת, מנסים שיטות שונות, רואים את התוצאות, ובהתאם בוחרים את השיטה שבעזרת מתקבלות התוצאות הטובות ביותר). אפשרות נוספת, שימוש ב-machine learning, שהולך ונהיה נפוץ יותר. אלו הן שיטות ממוחשבות שאפשר ליישם, שבודקות אפשרויות שונות, ובאופן אוטומטי יכולות לבחור את השיטה שתיתן את התוצאה הטובה ביותר.


Documents
Parsing
Linguistics
user query
Free text query parser
Results
page
Document
cache
Indexers
Spell correction
Scoring and ranking
Metadata in
zone and
field indexes
Inexact
top K
retrieval
Tiered inverted
positional index
k-gram
Indexes
Scoring
parameters
MLR
training
set