مبانی بازیابی اطلاعات و جستجوی وب

۶- Scoring, Term Weighting, The Vector Space Model

Original slides by: Hinrich Schütze and Christina Lioma

Overview

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Outline

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Ranked retrieval

- تا به اینجا تمامی پرس و جو ها دودویی بوده اند. اسناد یا تطابق دارند یا خیر.
- این نوع پرس و جو برای کاربران خبره مناسب است که نیازمندی ها خود و مجموعه اسناد را به صورت دقیق می شناسند.
- همچنین این پرس و جو برای برنامه های کاربردی مناسب است که میتوانند هزاران نتیجه را به سهولت پردازش کنند.
- اما برای اکثریت کاربران مناسب نیست.
- اکثر از کاربران توانایی تولید پرس و جوهای دودویی را ندارند، یا آن را تلاش مازاد میدانند.
- اکثر کاربران نمی خواهند که میان هزاران نتیجه (مخصوصا در جستجوی وب) به دنبال پاسخ خود بگردند.

4

Problem with Boolean search: Feast or famine

- پرس و جوهای دودویی معمولا منجر به تعداد نتایج بسیار کم یا بسیار زیاد می شوند.
- زمان زیادی نیاز است تا مهارت درج پرس و جو با تعداد نتایج قابل مدیریت را بدست آوریم.

- Query 1 (boolean conjunction): [standard user dlink 650]
  - → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
  - → 0 hits – famine

5

Feast or famine: No problem in ranked retrieval

- وقتی نتایج رتبه بندی شوند، مجموعه بزرگ نتایج مشکلی ایجاد نمی کند.
- فقط 10 نتیجه اول را نمایش می دهیم، بنابراین کاربر در میان نتایج غرق نمی شود!
- فرض این است که نتایج مرتبط تر نسبت به سایر نتایج رتبه بالاتری بدست می آورند.
- به هر جفت پرس و جو-سند امتیازی در بازه [0,1] می دهیم که نشان دهنده میزان تطابق سند و پرس و جو است.

6

Query-document matching scores

- How do we compute the score of a query-document pair?
- Let's start with a one-term query.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score
- We will look at a number of alternatives for doing this.

7

Take 1: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$(A \neq \emptyset \text{ or } B \neq \emptyset)$

- JACCARD $(A, A) = 1$
- JACCARD $(A, B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

8

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
  - Query: “ides of March”
  - Document “Caesar died in March”
  - JACCARD(q, d) = 1/6

9

What's wrong with Jaccard?

- تعداد رخداد یک کلمه را در سند در نظر نمی گیرد.
- لغات نادر اطلاعات بیشتری ارائه می دهند، اما معیار جاکارد این امر را در نظر نمی گیرد.

- We need a more sophisticated way of normalizing for the length of a document.
- Later in this lecture, we'll use $|A \cap B|/\sqrt{|A \cup B|}$ (cosine) . . .
- . . . instead of $|A \cap B|/|A \cup B|$ (Jaccard) for length normalization.

11

Outline

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

13

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

14

Bag of words model

- We do not consider the order of words in a document.
- *John is quicker than Mary and Mary is quicker than John* are represented the same way.
- This is called a bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- For now: bag of words model

15

Term frequency tf

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.

- قصد داریم از tf در زمان محاسبه تطابق پرس و جو-سند استفاده کنیم، چگونه؟
- رخداد خام کلمات کافی نیست زیرا:

- A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

16

Instead of raw frequency: Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$:
  $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms t in both *q* and *d*:
  tf-matching-score(q, d) = $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

17

Exercise

- Compute the Jaccard matching score and the tf matching score for the following query-document pairs.
- q: [information on cars] d: “all you’ve ever wanted to know about cars”
- q: [information on cars] d: “information on trucks, information on planes, information on trains”
- q: [red cars and red trucks] d: “cops stop red cars more often”

18

Outline

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Desired weight for rare terms

- علاوه بر فرکانس ترم ها (فرکانس رخداد ترم در سند)...
- ما به فرکانس ترم در مجموعه برای وزن دهی و رتبه بندی نیاز داریم

- کلمات نادر نسبت به کلمات پرتکرار اطلاعات بیشتری را منتقل می کنند
- Consider a term in the query that is rare in the collection
- A document containing this term is very likely to be relevant.
- → We want high weights for rare terms like ARACHNOCENTRIC.

20

Document frequency

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (positive) weights for frequent words like GOOD, INCREASE and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

21

idf weight

- df_t is the document frequency, the number of documents that *t* occurs in.
- df_t is an inverse measure of the informativeness of term *t*.
- We define the idf weight of term t as follows:

$$idf_t = \log_{10} \frac{N}{df_t}$$

(*N* is the number of documents in the collection.)

- idf_t is a measure of the informativeness of the term.
- [log N/df_t] instead of [N/df_t] to "dampen" the effect of idf
- Note that we use the log transformation for both term frequency and document frequency.

22

Examples for idf

- Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

23

Effect of idf on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query “arachnocentric line”, idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE.

24

Collection frequency vs. Document frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of *t*: number of tokens of *t* in the collection
- Document frequency of *t*: number of documents *t* occurs in
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df (and idf) is better for weighting than cf (and "icf").

25

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- Best known weighting scheme in information retrieval
- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

26

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document *d*:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

27

Exercise: Term, collection and document frequency

Quantity	Symbol	Definition
term frequency	$tf_{t,d}$	number of occurrences of *t* in *d*
document frequency	df_t	number of documents in the collection that *t* occurs in
collection frequency	cf_t	total number of occurrences of *t* in the collection

- Relationship between df and cf?
- Relationship between tf and cf?
- Relationship between tf and df?

28

Outline

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

30

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

31

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

Each document is now represented as a real-valued vector of tf idf weights $\in \mathbb{R}^{|V|}$.

32

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in \mathrm{R}^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

33

Queries as vectors

- نکته کلیدی اول: برای پرس و جو نیز به شیوه مشابه عمل کرده و آن را به صورت بردار در فضایی با ابعاد بالا نشان دهیم
- نکته کلیدی دوم: اسناد را با توجه به میزان شباهت به پرس و جو رتبه بندی نماییم.

- Recall: We're doing this because we want to get away from the you're-either-in-or-out, feast-or-famine Boolean model.
- Instead: rank relevant documents higher than nonrelevant documents

34

Use angle instead of distance

- Rank documents according to angle with query
- Thought experiment: take a document d and append it to itself. Call this document *d′*. *d′* is twice as long as *d*.
- "Semantically" *d* and *d′* have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

35

From angles to cosines

- The following two notions are equivalent.
  - Rank documents according to the angle between query and document in decreasing order
  - Rank documents according to cosine(query,document) in increasing order
- Cosine is a monotonically decreasing function of the angle for the interval [0°, 180°]

36

Cosine similarity between query and document

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

- q_i is the tf-idf weight of term *i* in the query.
- d_i is the tf-idf weight of term *i* in the document.
- $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$.
- This is the cosine similarity of $\vec{q}$ and $\vec{d}$ . . . . . or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

37

Cosine: Example

How similar are these novels? SaS: Sense and Sensibility PaP: Pride and Prejudice WH: Wuthering Heights

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

39

Cosine: Example

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

(To simplify this example, we don't do idf weighting.)

40

Cosine: Example

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

log frequency weighting & cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

- cos(SaS,PaP) ≈
  $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$
- cos(SaS,WH) ≈ 0.79
- cos(PaP,WH) ≈ 0.69
- Why do we have cos(SaS,PaP) > cos(SAS,WH)?

41

Optional: tf-idf example

- We often use different weightings for queries and documents.
- Notation: ddd.qqq
- Example: lnc.ltn
- document: logarithmic tf, no df weighting, cosine normalization
- query: logarithmic tf, idf, no normalization
- Isn't it bad to not idf-weight the document?
- Example query: "best car insurance"
- Example document: "car insurance auto insurance"

42

tf-idf example: lnc.ltn

Query: “best car insurance”. Document: “car insurance auto insurance”.

word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Key to columns: tf-raw: raw (unweighted) term frequency, tf-wght: logarithmically weighted term frequency, df: document frequency, idf: inverse document frequency, weight: the final weight of the term in the query or document, n’lized: document weights after cosine normalization, product: the product of final query weight and final document weight

$\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

1/1.92 ≈ 0.52

1.3/1.92 ≈ 0.68 Final similarity score between query and document: $\sum_i w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08$ Questions?

43

Summary: Ranked retrieval in the vector space model

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top *K* (e.g., *K* = 10) to the user

44

منابع

- فصل ششم و هفتم کتاب An introduction to information retrieval