CHAPTER TWO

PART II
Term Weighting and Similarity Measure

By Abio Ababor

June, 2021

Outline

- Bags of words
- Binary Weights
- Non-binary weights
  - Term Frequency
  - Document Frequency
  - Inverse Document Frequency
  - TF*IDF
- Similarity Measure
  - Euclidean distance
  - Inner Product
  - Cosine similarity

Recall the previous lecture

- What is text operation?
- Recall steps in text operation
- What is crawler?
- How it works?
- How real time stream of documents present for a user?
- What are the formats of documents?

Crawler

- **scale of the Web** —Trillions of pages distributed among billions of hosts.
- Another consideration is the **volume and variety of queries commercial Web search engines receive**
- Many pages **may change daily or hourly**.
  - **feeds**—Document feeds are a mechanism for accessing a real-time stream of documents. E.g. a news feed is a constant stream of news stories and updates.
  - Some content like **news, blogs, or videos are** used for web feeds. The reader monitors those feeds and provides new content when it arrives. **Radio** and **television** feeds are also used in some search applications, where the "documents" contain automatically segmented audio and video streams, together with associated text from closed captions or speech recognition.

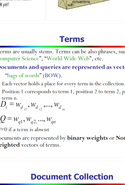
Crawler

- **Conversion:** The documents found by a crawler or provided by a feed are rarely in plain text. E.g. **HTML, XML, Adobe PDF, Microsoft Word, PPT** and so on, search engines require that these documents be converted into a consistent text plus metadata format.
- **Document data stored** on search engine database

- **Generally gathering copy of web pages from across web and storing locally for search engine for processing are the task of crawler.**

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'll always happy to see it again whenever I have a friend who hasn't seen it yet!



Terms

- **P** Terms are usually stems. Terms can be also phrases, such as "Computer Science", "World Wide Web", etc.
- **P** **Documents and queries are represented as vectors** or "bags of words" (BOW).
  - Each vector holds a place for every term in the collection.
  - Position i corresponds to term i, position 2 to term 2, position n to term n.

$$D = w_{1,d}, w_{2,d}, ..., w_{n,d}$$

$$Q = w_{1,q}, w_{2,q}, ..., w_{n,q}$$

If=0 if a term is absent

Documents are represented by **binary weights** or **Non-binary weighted** vectors of terms.

Document Collection

- A collection of n documents can be represented in the vector *space model* by a **term-document matrix**.
- An entry in the matrix corresponds to the "weight" of a term in the document; zero means the term has no significance in the document or it simply doesn't exist in the document.

$$\begin{array}{c} \\ D_1 \\ D_2 \\ \\ D_n \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \ldots & T_t \end{array} \\ \begin{array}{cccc} w_{11} & w_{21} & \ldots & w_{t1} \\ w_{12} & w_{22} & \ldots & w_{t2} \\ \vdots & \vdots & & \vdots \\ w_{1n} & w_{2n} & \ldots & w_{tn} \end{array} \end{array}$$

- Binary formula gives every word that appears in a document equal relevance.
- It can be useful when frequency is not important.
- **Binary Weights Formula:**

$$freq_{df} = \begin{cases} 1 & \text{if } freq_{df} > 0 \\ 0 & \text{if } freq_{df} = 0 \end{cases}$$

Binary Weights

	Doc1	Doc2	Doc3	Doc4
D1	1	0	0	1
D2	1	0	0	0
D3	1	0	0	2
D4	1	0	0	0
D5	1	0	0	1
D6	1	1	0	3
D7	0	1	0	0
D8	0	1	0	1
D9	0	0	1	1
D10	0	0	1	1
D11	1	0	1	1

Why use term weighting?

Binary weights are too limiting.

- terms are either present or absent.
- Not allow to order documents according to their level of relevance for a given query

Non-binary weights allow to model partial matching.

- Partial matching allows retrieval of docs that approximate the query.

Term-weighting improves quality of answer set.

- Term weighting enables ranking of retrieved documents; such that best matching documents are ordered at the top as they are more relevant than others.

Term Weighting: Term Frequency (TF)

- TF (term frequency) - Count the number of times term occurs in document.
  - $f_{t,d}$ = **frequency of term t in document d**.
- The more times a term t occurs in document d the more likely it is that t is relevant to the document, i.e. more indicative of the topic.
- If used alone, it favors common words and long documents.
- It gives too much credit to words that appears more frequently.
- May want to normalize *term frequency* (tf) across the entire corpus: $tf_{t,d} = f_{t,d} / \sum_t f_{t,d}$

docs	t1	t2	t3
D1	2	0	3
D2	1	0	0
D3	0	4	7
D4	3	0	0
D5	1	6	3
D6	3	5	9
D7	0	8	0
D8	0	10	0
D9	0	0	1
D10	0	3	5
D11	4	0	1

Document Normalization

- **Long documents have an unfair advantage:**
  - They use a lot of terms
    - So they get more matches than short documents
  - And they use the same words repeatedly
    - So they have much higher term frequencies
- **Normalization seeks to remove these effects:**
  - Related somehow to maximum term frequency.
  - But also sensitive to the total number of terms.

If we don't normalize short documents may not be recognized as relevant.

Problems with term frequency

- We need a mechanism for reducing the effect of terms that occur too often in the collection to be meaningful for relevance/meaning determination
- Scale down the term weight of terms with high collection frequency
  - Reduce the tf weight of a term by a factor that grows with the collection frequency
- More common for this purpose is document frequency
  - how many documents in the collection contain the term
- The example shows that collection frequency and document frequency behaves differently

Word	cf	df
try	10524	8700
insurance	10440	3997

Document Frequency

It is defined to be the number of documents in the collection that contain a term

DF = document frequency

- Count the frequency considering the whole collection of documents.
- Less frequently a term appears in the whole collection, the more discriminating it is.

df_t = document frequency of term t
= number of documents containing term t

Inverse Document Frequency (IDF)

IDF measures rarity of the term in collection.
The IDF is a measure of the general importance of the term

- It is the inverse of the document frequency.
- It diminishes the weight of terms that occur very frequently in the collection and increases the weight of terms that occur rarely.
- Gives full weight to terms that occur in one document only.
- Gives lowest weight to terms that occur in all documents.
- Terms that appear in many *different* documents are *less* indicative of overall topic.

idf_t = inverse document frequency of term t,
$= \log_e (N / df_t)$ (N: total number of documents)

Inverse Document Frequency

- E.g. given a collection of 1000 documents and document frequencies, compute IDF for each word:

Word	N	DF	IDF
the	1000	1000	0
some	1000	100	3.322
car	1000	10	6.644
merge	1000	1	9.966

- IDF provides high values for rare words and low values for common words.
- IDF is an indication of a term's *discrimination power*.
  - Log used to dampen the effect relative to tf.
  - Make the difference between Document frequency vs. corpus frequency ?

TF*IDF Weighting

The most used term-weighting is $tf*idf$ *weighting scheme*: $w_{ij} = tf_{ij}\, idf_i = tf_{ij} * \log_2 (N / df_i)$

A term occurring frequently in the document but rarely in the rest of the collection is given high weight.

- The tf-idf value for a term will always be greater than or equal to zero.

Experimentally, $tf*idf$ has been found to work well.

- *It is often used in the vector space model together with cosine similarity to determine the similarity between two documents.*

TF*IDF weighting

When does $tf*idf$ registers a high weight? when a term t occurs many times within a small number of documents

- Highest $tf*idf$ for a term shows a term has a *high term frequency (in the given document) and a few document frequency* (in the whole collection of documents);
- the weights hence tend to filter out common terms.
  - Thus lending high discriminating power to those documents
- Lower TF*IDF is registered when the term occurs fewer times in a document, or occurs in many documents
- Lower TF*IDF is registered when the term occurs in virtually all documents

Computing TF-IDF: An Example

Assume collection contains 10,000 documents and statistical analysis shows that document frequencies (DF) of three terms are: A(50), B(1300), C(250). And also term frequencies (TF) of these terms are: A(3), B(23), C(1). Compute TF*IDF for each term?

A: tf = 3/3=1.00; idf = $\log_2$(10000/ 50) = 7.644; tf*idf = 7.644
B: tf = 23/ 100 =0.67; idf = $\log_2$(10000 / 1300) = 2.943; tf*idf = 1.962
C: tf = 1/ 100 =0.01; idf = $\log_2$(10000 / 250) = 5.322; tf*idf = 0.053

$C_1 = 0 + 0 + 0.43 = 0.43$; $C_2 = 0 + 0 + 0 + 0.43 + 0.43 = 0.86$; $C_3 = 0.43 + 0.43 + 0.43 + 0.43 = 1.72$

Query vector is typically treated as a document and also tf-idf weighted.

More Example

- Consider a document containing 100 words wherein the word *cow* appears 3 times. Now, assume we have 10 million documents and *cow* appears in one thousand of these.
  - The term frequency (TF) for *cow* :
    $3/100 = 0.03$
  - The inverse document frequency is
    $\log_{10}(10{,}000{,}000 \ / \ 1{,}000) = 13.228$
  - The TF*IDF score is the product of these frequencies: 0.03 * 13.228 = 0.39684

Exercise

- Let C = number of times a given word appears in a document;
- TW = total number of words in a document;
- TD = total number of documents in a corpus, and
- DF = total number of documents containing a given word;
- ***compute TF, IDF and TF*IDF score for each term***

Word	C	TW	TD	DF	TF	IDF	TFIDF
airplane	5	45	3	1			
blue	1	45	3	1			
chair	7	45	3	3			
computer	3	45	3	1			
forest	2	45	3	1			
justice	7	45	3	3			
love	2	45	3	1			
might	2	45	3	1			
part	5	45	3	2			
rose	6	45	3	3			
shoe	4	45	3	1			
thesis	2	45	3	2			

Exercises

- A database collection consists of 1 million documents, of which 200,000 contain the term holiday while 250,000 contain the term season. A document repeats holiday 7 times and season 5 times. It is known that holiday is represented more than any other term in the document. Calculate the TF*IDF score for this document using three different term weight methods. Try with
  (i) normalized and unnormalized *TF*;
  (ii) *TF*IDF based on* normalized and unnormalized *TF*
  N=1000000
  Holiday: 1 df=200,000
  season: 1 250,000
  Max fw=7
  Unnm TFholiday=7 unr =7/7=1
  Unnr tfSeason=5 unr 5/7=
  *tf-idf=tf*log2(N/df)*
  *tf-idf Season =5*log2(1000000/250000)=5**
  *(5/7)*log(1000000/250000)=……*

Concluding remarks

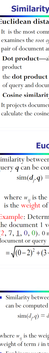
Hence IDF is incorporated which diminishes the weight of terms that occur very frequently in the collection and increases the weight of terms that occur rarely.

- This leads to use TF*IDF as a better weighting technique
- On top of that we apply similarity measures to calculate the distance between document i and query j.
- There are a number of similarity measures; the most common similarity measure are
  - Euclidean distance , Inner or Dot product, Cosine similarity, Dice similarity, Jaccard similarity, etc.

Similarity Measure

- We now have vectors for all documents in the collection, and a vector for the query.
- How do we compute similarity?
- A similarity measure is a function that computes the *degree of similarity* or distance between document vector and query vector.
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of presumed relevance.
  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Intuition



Postulate: Documents that are "close together" in the vector space talk about the same things and are more similar than others.

Similarity Measure

If d_1 is near d_2, then d_2 is near d_1.
If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
No document is closer to d than d itself.

- Sometimes it is a good idea to determine the maximum possible similarity as the "distance" between a document d and itself.

A similarity measure attempts to compute the distance between document vector w_i and query w_q vector.

- The assumption here is that documents whose vectors are close to the query vector are more relevant to the query than documents whose vectors are away from the query vector.

Similarity Measure: Techniques

- **Euclidean distance**
  - It is the most common similarity measure. Euclidean distance examines the *root of square differences* between coordinates of a pair of document and query terms.
- **Dot product**—also known as the scalar product or inner product
  - the **dot product** is defined as the product of the magnitudes of query and document vectors
- ***Cosine similarity*** (or normalized *inner product*)
  - It projects document and query vectors into a term space and calculate the cosine angle between these.

Euclidean distance

Similarity between vectors for the document d_i and query q can be computed as:

$$sim(d_i,q) = |d_i - q| = \sqrt{\sum_{t=1}^{n} (w_{ti} - w_{tq})^2}$$

where w_{ti} is the weight of term t in document j and w_{tq} is the weight of term t in the query q

- Example: Determine the Euclidean distance between the document 1 vector (0, 3, 2, 1, 10) and query vector (1, 7, 1, 0, 0). *0 means corresponding term not found in document or query*

$$= \sqrt{(0-2)^2 + (3-7)^2 + (2-1)^2 + (1-0)^2 + (10-0)^2} = 11.05$$

Inner Product

- Similarity between vectors for the document d_i and query q can be computed as the vector inner product:

$$sim(d_i,q) = d_i \bullet q = \quad w_{ti} \bullet w_{tq} = \sum_{t=1}^{n} w_{ti} \bullet q_{ti}$$

where w_{ti} is the weight of term i in document j and w_{tq} is the weight of term i in the query q

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).
- For weighted term vectors, it is the sum of the products of the weights of the matched terms.
- Measures how many terms matched but not how many terms are *not* matched.

Inner Product -- Examples

Binary weight :

- Size of vector = size of vocabulary = 7 sim(D, Q) = 3

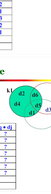
	Retrieval	Database	Term	Computer	Text	Manage	Data
D 1	1	1	1	0	1	1	0
Q 1	0	1	1	0	0	1	1

- **Term Weighted:**

	Retrieval	Database	Architecture
D1	2	3	5
D2	3	7	1
Q	0	0	2

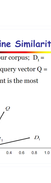
sim(D1 , Q) = 2 × 3*0 + 3 *2 = 10
sim(D2 , Q) = 3*0 + 7*0 + 1*2 = 2

Inner Product: Example 1



	k1	k2	k3	q • dj
d1	1	0	0	1
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	2	3	

Inner Product: Exercise



	k1	k2	k3	q • dj
d1	1	0	1	?
d2	1	0	1	?
d3	0	1	1	?
d4	1	0	0	?
d5	1	1	1	?
d6	1	1	0	?
d7	0	1	0	?
q	1	2	3	

Cosine similarity

- Measures similarity between d_j and d_q *captured by* the cosine of the angle θ between them.

$$sim(d_j,q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \bullet \left|\vec{q}\right|} = \frac{\sum_{i=1}^{n} w_{ij} \bullet w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \bullet \sqrt{\sum_{i=1}^{n} w_{iq}^2}}$$

Or:

$$sim(d_j,q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \bullet \left|\vec{q}\right|} = \frac{\sum_{i=1}^{n} w_{ij} \bullet w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \bullet \sqrt{\sum_{i=1}^{n} w_{iq}^2}}$$

- The denominator involves the lengths of the vectors
- So the cosine measure is also known as the *normalized inner product*

$$\text{Length}\left|\vec{d_j}\right| = \sqrt{\sum_{i=1}^{n} w_{ij}^2}$$

Example: Computing Cosine Similarity

- Let us say we have a query vector Q = (0.4, 0.8); and a document vector D_1 = (0.2, 0.7). Compute their similarity using cosine?

$$sim(Q,D_1) = \frac{(0.4 * 0.2) + (0.8 * 0.7)}{\sqrt{(0.4)^2 + (0.8)^2} * \sqrt{(0.2)^2 + (0.7)^2}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

Example: Computing Cosine Similarity

- Let say we have two documents in our corpus: D_1 = (0.8, 0.3) and D_2 = (0.2, 0.7). Given query vector Q = (0.4, 0.8), determine which document is the most relevant for the query?

$$\cos\theta_1 = 0.73$$
$$\cos\theta_2 = 0.98$$

Example

- Given three documents; D_1, D_2 and D_3 with the corresponding TFIDF weight, Which documents are more similar using the three measurement?

Terms	D1	D2	D3
affection	0.996	0.993	0.847
Jealous	0.087	0.120	0.466
gossip	0.017	0.000	0.254

Cosine Similarity vs. Inner Product

- Cosine similarity measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.

$$\text{CosSim}(\vec{d_j}, \vec{q}) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \left|\vec{q}\right|} = \frac{\sum_{i=1}^{n} (w_{ij} \bullet w_{iq})}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \ \sqrt{\sum_{i=1}^{n} w_{iq}^2}}$$

$$\text{Inner Product}(\vec{d_j}, \vec{q}) = \vec{d_j} \bullet \vec{q}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1, Q) = 10 / $\sqrt{4+9+25}$*$\sqrt{0+0+4}$ = 0.81
$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2, Q) = 2 / $\sqrt{9+49+1}$*$\sqrt{0+0+4}$=0.13
$Q = 0T_1 + 0T_2 + 2T_3$

D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product. Red color(10 and 2) show inner product

Thank you

Thank you