# SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## COURSE NAME : 19CS732 INFORMATION RETRIEVAL TECHNIQUES

IVYEAR / VIII SEMESTER

Unit 2- MODELING AND RETRIEVAL EVALUATION

Topic 3 : TF-IDF (Term Frequency/Inverse Document Frequency) Weighting

# Problem

- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression, which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
- The model frequently returns either too few or too many documents in response to a user query

# TF-IDF (Term Frequency/Inverse Document Frequency) Weighting

- Vocabulary
- Stop word lists

Commonly occurring words are unlikely to give useful information and may be removed from the vocabulary to speed processing

Stopword lists contain frequent words to be excluded

Stopword lists need to be used carefully

E.g. “to be or not to be”

# TF-IDF (Term Frequency/Inverse Document Frequency) Weighting-Cont..

- Term weighting

Not all words are equally useful

A word is most likely to be highly relevant to document A if it is:

- Infrequent in other documents
- Frequent in document A

# Normalised term frequency (tf)

- A normalised measure of the importance of a word to a document is its frequency, divided by the maximum frequency of any term in the document
- This is known as the tf factor.
- Document A: raw frequency vector: (2,1,1,1,0), tf vector: (  )
- This stops large documents from scoring higher

# Inverse document frequency (idf)

A calculation designed to make rare words more important than common words

The idf of word *i* is given by

$$idf_i = \log \frac{N}{n_i}$$

Where $N$ is the number of documents and $n_i$ is the number that contain word $i$

# TF-IDF

- The tf-idf weighting scheme is to multiply each word in each document by its tf factor and idf factor
- Different schemes are usually used for query vectors
- Different variants of tf-idf are also used

# A FIRST TAKE AT BUILDING AN INVERTED INDEX

**A first take at building an inverted index**

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance. The major steps in this are:

1.Collect the documents to be indexed:

Friends, Romans, countrymen. So let it be with Caesar ....

2.Tokenize the text, turning each document into a list of tokens:

Friends Romans countrymen So ... countryman

3.Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms:

4.Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

# A First Take at Building an Inverted Index -Cont..

**Doc 1**
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

**Doc 2**
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

| term | docID |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |

⟹

| term | docID |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |

⟹

| term | doc. freq. | → | postings lists |
|---|---|---|---|
| ambitious | 1 | → | 2 |
| be | 1 | → | 2 |
| brutus | 2 | → | 1 → 2 |
| capitol | 1 | → | 1 |
| caesar | 2 | → | 1 → 2 |
| did | 1 | → | 1 |
| enact | 1 | → | 1 |
| hath | 1 | → | 2 |
| I | 1 | → | 1 |
| i' | 1 | → | 1 |
| it | 1 | → | 2 |
| julius | 1 | → | 1 |
| killed | 1 | → | 1 |
| let | 1 | → | 2 |
| me | 1 | → | 1 |
| noble | 1 | → | 2 |
| so | 1 | → | 2 |
| the | 2 | → | 1 → 2 |
| told | 1 | → | 2 |
| you | 1 | → | 2 |
| was | 2 | → | 1 → 2 |
| with | 1 | → | 2 |

► **Figure 1.3** Building an index by sorting and grouping. The sequence of terms in each document, tagged by their documentID (left) is sorted alphabetically (middle). Instances of the same term are then grouped by word and then by documentID. The terms and documentIDs are then separated out (right). The dictionary stores the terms, and has a pointer to the postings list for each term. It commonly also stores other summary information such as, here, the document frequency of each term. We use this information for improving query time efficiency and, later, for weighting in ranked retrieval models. Each postings list stores the list of documents in which a term occurs, and may store other information such as the term frequency (the frequency of each term in each document) or the position(s) of the term in each document.

# A First Take at Building an Inverted Index -Cont..

Draw the inverted index that would be built for the following document collection. (See Figure 1.3 for an example.)

**Doc 1** new home sales top forecasts
**Doc 2** home sales rise in july
**Doc 3** increase in home sales in july
**Doc 4** july new home sales rise

Consider these documents:**Doc 1** breakthrough drug for schizophrenia
**Doc 2** new schizophrenia drug
**Doc 3** new approach for treatment of schizophrenia
**Doc 4** new hopes for schizophrenia patients

Draw the term-document incidence matrix for this document collection.

Draw the inverted index representation for this collection, as in Figure 1.3 (page ).

# TF-IDF

- The tf-idf weighting scheme is to multiply each word in each document by its tf factor and idf factor
- Different schemes are usually used for query vectors
- Different variants of tf-idf are also used

# TF- Weighting

## 2.3 Term frequency and weighting

- We assign to each term in a document a *weight* for that term that depends on the number of occurrences of the term in the document.
- We would like to compute a score between a query term $t$ and a document $d$, based on the weight of $t$ in $d$. The simplest approach is to assign the weight to be equal to the number of occurrences of term $t$ in document $d$.
- This weighting scheme is referred to as term frequency and is denoted $tf_{t,d}$, with the subscripts denoting the term and the document in order.
- For a document $d$, the set of weights determined by the tf weights above (or indeed any weighting function that maps the number of occurrences of $t$ in $d$ to a positive real value) may be viewed as a quantitative digest of that document.
- In this view of a document, known in the literature as the *bag of words model*, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material (in contrast to Boolean retrieval).

## Inverse document frequency

- Raw term frequency as above suffers from a critical problem: all terms are considered equally important when it comes to assessing relevancy on a query.
- For instance, a collection of documents on the auto industry is likely to have the term auto in almost every document. To this end, we introduce a mechanism for attenuating the effect of terms that occur too often in the collection to be meaningful for relevance determination.
- An immediate idea is to scale down the term weights of terms with high *collection frequency*, defined to be the total number of occurrences of a term in the collection.
- The idea would be to reduce the tf weight of a term by a factor that grows with its collection frequency. Instead, it is more commonplace to use for this purpose the *document frequency dft*, defined to be the number of documents in the collection that contain a term t.
- This is because in trying to discriminate between documents for the purpose of scoring it is better to use a document-level statistic (such as the number of documents containing a term) than to use a collection-wide statistic for the term.

# TF- Weighting-Cont..

- The terms of a document are not equally useful for describing the document contents
- In fact, there are index terms which are simply vaguer than others
- There are properties of an index term which are useful for evaluating the importance of the term in a document
- For instance, a word which appears in all documents of a collection is completely useless for retrieval tasks

# TF- Weighting-Cont..



- The reason to prefer df to cf is illustrated in Figure 2.3, where a simple example shows that collection frequency (cf) and document frequency (df) can behave rather differently.In particular, the cf values for both try and insurance are roughly equal, but their df values differ significantly.
- Intuitively, we want the few documents that contain insurance to get a higher boost for a query on insurance than the many documents containing try get from a query on try.

| Word | cf | df |
|---|---|---|
| try | 10422 | 8760 |
| insurance | 10440 | 3997 |

**Figure 2.3** Collection frequency (cf) and document frequency (df) behave differently, as in this example from the Reuters collection.

How is the document frequency df of a term used to scale its weight? Denoting as usual the total number of documents in a collection by $N$, we define the *inverse document frequency* (idf) of a term $t$ as follows:

$$\text{idf}_t = \log$$

Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low. Figure 2.4 gives an example of idf's in the Reuters collection of 806,791 documents; in this example logarithms are to the base 10.

| Term | $df_t$ | $idf_t$ |
|---|---|---|
| car | 18,165 | 1.65 |
| auto | 6723 | 2.08 |
| insurance | 19,241 | 1.62 |
| best | 25,235 | 1.5 |

**Figure 2.4** Example of idf values. Here we give the idf's of terms with various frequencies in the Reuters collection of 806,791 documents.

# TF- Weighting-Cont..

**Tf-idf weighting**

- We now combine the definitions of term frequency and inverse document frequency, to produce a composite weight for each term in each document.
- The *tf-idf* weighting scheme assigns to term $t$ a weight in document $d$ given by

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

- In other words, $\text{tf-idf}_{t,d}$ assigns to term $t$ a weight in document $d$ that is

1. Highest when $t$ occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. Lowest when the term occurs in virtually all documents.

- At this point, we may view each document as a *vector* with one component corresponding to each term in the dictionary, together with a weight for each component that is given by equation above. For dictionary terms that do not occur in a document, this weight is zero.
- Document $d$ is the sum, over all query terms, of the number of times each of the query terms occurs in $d$.
- We can refine this idea so that we add up not the number of occurrences of each query term $t$ in $d$, but instead the tf-idf weight of each term in $d$.

Score $(q, d)$ =

**Cosine similarity**

- Documents could be ranked by computing the distance between the points representing the documents and the query.
- More commonly, a ***similarity measure*** is used (rather than a distance or *dissimilarity* measure), so that the documents with the highest scores are the most similar to the query.
- A number of similarity measures have been proposed and tested for this purpose.
- The most successful of these is the *cosine correlation* similarity measure.

# TF- Weighting-Cont..

- The cosine correlation measures the cosine of the angle between the query and the document vectors.
- When the vectors are *normalized* so that all documents and queries are represented by vectors of equal length, the cosine of the angle between two identical vectors will be 1 (the angle is zero), and for two vectors that do not share any non-zero terms, the cosine will be 0.
- The cosinc measure is defined as:

$$Cosine\ (Di,\ Q) = \frac{\Sigma'_{j=1} d_{ij}\ q_j}{\sqrt{\Sigma'_{j=1} d_{ij}\ 2.\Sigma'_{j=1}\ q_j 2}}$$

- The numerator of this measure is the sum of the products of the term weights for the matching query and document terms (known as the *dot product* or inner product).
- The denominator normalizes this score by dividing by the product of the lengths of the two vectors. There is no theoretical reason why the cosine correlation should be preferred to other similarity measures, but it does perform somewhat better in evaluations of search quality.
- As an example, consider two documents $D1 = (0.5,\ 0.8,\ 0.3)$ and $D2 = (0.9,\ 0.4,\ 0.2)$ indexed by three terms, where the numbers represent term weights.
- Given the query $Q = (1.5,\ 1.0,\ 0)$ indexed by the same terms, the cosine measures for the two documents are:

$$Cosine\ (D1,\ Q) = \frac{(0.5\times1.5)+(0.8\times1.0)}{\sqrt{(0.52+0.82+0.32)(1.52+1.02)}} = 0.87$$

$$Cosine\ (D2,\ Q) = \frac{(0.9\times1.5)+(0.4\times1.0)}{\sqrt{(0.92+0.42+0.22)(1.52+1.02)}} = 0.97$$

- The second document has a higher score because it has a high weight for the first term, which also has a high weight in the query.
- Even this simple example shows that ranking based on the vector space model is able to reflect term importance and the number of matching terms, which is not possible in Boolean retrieval.

# Activity

# Disadvantages

- assumes independence of index terms; not clear that this is bad though

# Advantages

- term-weighting improves answer set quality
- partial matching allows retrieval of docs that approximate the query conditions
- cosine ranking formula sorts documents according to degree of similarity to the query

# Assessment 1

1. List out the Advantages of TF-IDF
   a)_____
   b)_____
   c)_____
   d)_____

2. Identify the disadvantages of TF-IDF
   a)_____
   b)_____
   c)_____
   d)_____

**TEXT BOOKS:**

1. Ricardo Baeza-Yates and Berthier Ribeiro-Neto, —Modern Information Retrieval: The Concepts and Technology behind Search, Second Edition, ACM Press Books, 2011.

2. Ricci, F, Rokach, L. Shapira, B.Kantor, —Recommender Systems Handbook||, First Edition, 2011.

**REFERENCES:**

1. C. Manning, P. Raghavan, and H. Schütze, —Introduction to Information Retrieval, Cambridge University Press, 2008.

2. Stefan Buettcher, Charles L. A. Clarke and Gordon V. Cormack, —Information Retrieval: Implementing and Evaluating Search Engines, The MIT Press, 2010.

# THANK YOU