Introduction to Information Retrieval
http://informationretrieval.org

IIR 6: Scoring, Term Weighting, The Vector Space Model

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2008.05.20
Overview

1. Recap
2. Term frequency
3. tf-idf weighting
4. The vector space
Outline

1. Recap
2. Term frequency
3. tf-idf weighting
4. The vector space
Heaps’ law

Vocabulary size $M$ as a function of collection size $T$ (number of tokens) for Reuters-RCV1. For these data, the dashed line $\log_{10} M = 0.49 \times \log_{10} T + 1.64$ is the best least squares fit. Thus, $M = 10^{1.64} T^{0.49}$ and $k = 10^{1.64} \approx 44$ and $b = 0.49$. 

![Graph showing logarithmic relationship between vocabulary size and collection size](image)
Zipf’s law

- Zipf’s law: The $i^{th}$ most frequent term has frequency proportional to $1/i$.
- $c_f_i \propto \frac{1}{i}$
- $c_f$ is collection frequency: the number of occurrences of the term in the collection.
- So if the most frequent term ($the$) occurs $c_f_1$ times, then the second most frequent term ($of$) has $c_f_1/2$ occurrences, ...
- ...the third most frequent term ($and$) has $c_f_1/3$ occurrences etc.
- About half of all vocabulary terms occur only once in the collection. (hapax legomena)
- Zipf’s law is an example of a power law.
Dictionary as a string with blocking

\[\ldots 7\text{style} 9\text{syzygetic} 8\text{syzygial} 6\text{syzygy} 11\text{szabelyite} 6\text{szecin}\ldots\]

freq. postings ptr. term ptr.

\[
\begin{align*}
9 & \to \\
92 & \to \\
5 & \to \\
71 & \to \\
12 & \to \\
\ldots & \ldots & \ldots & \ldots
\end{align*}
\]
Variable byte (VB) code

- Dedicate 1 bit (high bit) to be a **continuation bit** $c$.
- If the gap $G$ fits within 7 bits, binary-encode it in the 7 available bits and set $c = 1$.
- Else: set $c = 0$, encode high-order 7 bits and then use one or more additional bytes to encode the lower order bits using the same algorithm.
Gamma codes for gap encoding

- You can get even more compression with \textit{bitlevel} code.
- Gamma code is the best known of these.
- Represent a gap $G$ as a pair of length and offset.
- Offset is the gap in binary, with the leading bit chopped off.
- For example $13 \rightarrow 1101 \rightarrow 101$
- Length is the length of offset.
- For 13 (offset 101), this is 3.
- Encode length in \textit{unary} code: 1110.
- Gamma code of 13 is the concatenation of length and offset: 1110101.
Outline

1. Recap
2. Term frequency
3. tf-idf weighting
4. The vector space
Ranked retrieval

- Thus far, our queries have all been Boolean.
Ranked retrieval

Thus far, our queries have all been Boolean.
Documents either match or don’t.
Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
- Also good for applications: Applications can easily consume 1000s of results.
Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
 Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don't.
- Good for expert users with precise understanding of their needs and the collection.
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
- Most users are not capable of writing Boolean queries (or they are, but they think it's too much work).
Ranked retrieval

• Thus far, our queries have all been Boolean.
  • Documents either match or don’t.
• Good for expert users with precise understanding of their needs and the collection.
• Also good for applications: Applications can easily consume 1000s of results.
• Not good for the majority of users.
• Most users are not capable of writing Boolean queries (or they are, but they think it’s too much work).
• Most users don’t want to wade through 1000s of results.
Ranked retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don’t.
- Good for expert users with precise understanding of their needs and the collection.
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
- Most users are not capable of writing Boolean queries (or they are, but they think it’s too much work).
- Most users don’t want to wade through 1000s of results.
- This is particularly true of web search.
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (\(=0\)) or too many (1000s) results.
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: “standard user dlink 650” → 200,000 hits
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (≈0) or too many (1000s) results.
- Query 1: “standard user dlink 650” → 200,000 hits
- Query 2: “standard user dlink 650 no card found”: 0 hits
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: “standard user dlink 650” → 200,000 hits
- Query 2: “standard user dlink 650 no card found”: 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (0) or too many (1000s) results.
- Query 1: “standard user dlink 650” → 200,000 hits
- Query 2: “standard user dlink 650 no card found”: 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
- With a ranked list of documents it does not matter how large the retrieved set is.
Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher.
Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher.
- How can we rank-order the documents in the collection with respect to a query?
Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher.
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in [0, 1] – to each document
Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher.
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in [0, 1] – to each document
- This score measures how well document and query “match”.

Schütze: Scoring, term weighting, the vector space model
Query-document matching scores

- We need a way of assigning a score to a query/document pair.
Query-document matching scores

- We need a way of assigning a score to a query/document pair.
- Let’s start with a one-term query.
Query-document matching scores

- We need a way of assigning a score to 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.
Query-document matching scores

- We need a way of assigning a score to 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.
Query-document matching scores

- We need a way of assigning a score to 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.
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let $A$ and $B$ be two sets
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let $A$ and $B$ be two sets
- Jaccard coefficient:

$$JACCARD(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let $A$ and $B$ be two sets
- Jaccard coefficient:

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

- $\text{JACCARD}(A, A) = 1$
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let \( A \) and \( B \) be two sets
- Jaccard coefficient:

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

- \( \text{JACCARD}(A, A) = 1 \)
- \( \text{JACCARD}(A, B) = 0 \) if \( A \cap B = 0 \)
Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let $A$ and $B$ be two sets
- Jaccard coefficient:

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

- $\text{JACCARD}(A, A) = 1$
- $\text{JACCARD}(A, B) = 0$ if $A \cap B = 0$
- $A$ and $B$ don’t have to be the same size.
Recap

Term frequency tf-idf weighting

The vector space

Take 1: Jaccard coefficient

- Recall from IIR 3: A commonly used measure of overlap of two sets
- Let $A$ and $B$ be two sets
- Jaccard coefficient:

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

- $\text{JACCARD}(A, A) = 1$
- $\text{JACCARD}(A, B) = 0$ if $A \cap B = 0$
- $A$ and $B$ don’t have to be the same size.
- Always assigns a number between 0 and 1.
What is the query-document match score that the Jaccard coefficient computes for:
Jaccard coefficient: Example

What is the query-document match score that the Jaccard coefficient computes for:

- Query: “ides of March”
What is the query-document match score that the Jaccard coefficient computes for:

- Query: “ides of March”
- Document “Caesar died in March”
What is the query-document match score that the Jaccard coefficient computes for:

- Query: “ides of March”
- Document “Caesar died in March”
- ?
What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard doesn’t consider this information.
What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard doesn’t consider this information.
- We need a more sophisticated way of normalizing for length.
What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard doesn’t consider this information.
- We need a more sophisticated way of normalizing for length.
- Later in this lecture, we’ll use $|A \cap B|/\sqrt{|A \cup B|}$ (cosine) . . .
What’s wrong with Jaccard?

- It doesn’t consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard doesn’t consider this information.
- We need a more sophisticated way of normalizing for length.
- 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.
Recall: Binary incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony</th>
<th>Julius</th>
<th>The</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Worser</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Each document is represented by a binary vector $\in \{0, 1\}^{|V|}$. 
Recall: Binary incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Worse</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Each document is represented by a binary vector $\in \{0, 1\}^{|V|}$. 
From now on, we will use the frequencies of terms

|       | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ...
|-------|------------------------|---------------|-------------|--------|---------|---------|-----
| Antony | 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 represented by a count vector $\in \mathbb{N}^{||V||}$. 
From now on, we will use the frequencies of terms

| Term  | Anthony | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony and Cleopatra</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Worser</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Each document is represented by a count vector $\in \mathbb{N}^{|V|}$. 
Bag of words model

- We do not consider the order of words in a document.
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.
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.
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.
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.
- We will look at “recovering” positional information later in this course.
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.
- We will look at “recovering” positional information later in this course.
- For now: bag of words model
The term frequency $tf_{t,d}$ of term $t$ in document $d$ is defined as the number of times that $t$ occurs in $d$. 
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 $d$.
- We want to use $tf$ when computing query-document match scores.
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 $d$**.
- We want to use $tf$ when computing query-document match scores.
- But how?
Term frequency $t_f$ 

- The term frequency $t_{f,t,d}$ of term $t$ in document $d$ is defined as the number of times that $t$ occurs in $d$.
- We want to use $t_f$ when computing query-document match scores.
- But how?
- Raw term frequency is not what we want.
The term frequency $tf_{t,d}$ of term $t$ in document $d$ is defined as the **number of times that $t$ occurs in $d$**.

We want to use $tf$ when computing query-document match scores.

But how?

Raw term frequency is not what we want.

A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.
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 $d$.
- We want to use $tf$ when computing query-document match scores.
- But how?
- Raw term frequency is not what we want.
- A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.
- But not 10 times more relevant.
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 $d$.
- We want to use $tf$ when computing query-document match scores.
- But how?
- Raw term frequency is not what we want.
- A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.
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 $d$.
- We want to use $tf$ when computing query-document match scores.
- But how?
- Raw term frequency is not what we want.
- A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.
The term frequency $tf_{t,d}$ of term $t$ in document $d$ is defined as the number of times that $t$ occurs in $d$.

We want to use $tf$ when computing query-document match scores.

But how?

Raw term frequency is not what we want.

A document with 10 occurrences of the term is more relevant than a document with one occurrence of the term.

But not 10 times more relevant.

Relevance does not increase proportionally with term frequency.
The log frequency weight of term $t$ in $d$ is defined as follows

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

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

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

- $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
The log frequency weight of term \( t \) in \( d \) is defined as follows:

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

- \( 0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4 \), etc.
- Score for a document-query pair: sum over terms \( t \) in both \( q \) and \( d \):
  \[
  \text{matching-score} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})
  \]
Log frequency weighting

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

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

- $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.

- Score for a document-query pair: sum over terms $t$ in both $q$ and $d$:

$$\text{matching-score} = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$$

- The score is 0 if none of the query terms is present in the document.
Outline

1 Recap
2 Term frequency
3 tf-idf weighting
4 The vector space
Document frequency

- Rare terms are more informative than frequent terms.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is *rare* in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
  - → We want a high weight for rare terms like ARACHNOCENTRIC.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
  - → We want a high weight for rare terms like ARACHNOCENTRIC.
- Consider a term in the query that is frequent in the collection (e.g., HIGH, INCREASE, LINE)
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
  - → We want a high weight for rare terms like ARACHNOCENTRIC.
- Consider a term in the query that is frequent in the collection (e.g., HIGH, INCREASE, LINE)
  - A document containing this term is more likely to be relevant than a document that doesn't, but it's not a sure indicator of relevance.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
  - → We want a high weight for rare terms like ARACHNOCENTRIC.
- Consider a term in the query that is frequent in the collection (e.g., HIGH, INCREASE, LINE)
  - A document containing this term is more likely to be relevant than a document that doesn't, but it’s not a sure indicator of relevance.
  - → For frequent terms, we want positive weights for words like HIGH, INCREASE, and LINE, but lower weights than for rare terms.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., \textsc{arachnocentric})
  - A document containing this term is very likely to be relevant.
  - $\rightarrow$ We want a high weight for rare terms like \textsc{arachnocentric}.
- Consider a term in the query that is frequent in the collection (e.g., \textsc{high, increase, line})
  - A document containing this term is more likely to be relevant than a document that doesn't, but it's not a sure indicator of relevance.
  - $\rightarrow$ For frequent terms, we want positive weights for words like \textsc{high, increase, line}, but lower weights than for rare terms.
- We will use document frequency to factor this into computing the matching score.
Document frequency

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC)
  - A document containing this term is very likely to be relevant.
  - → We want a high weight for rare terms like ARACHNOCENTRIC.
- Consider a term in the query that is frequent in the collection (e.g., HIGH, INCREASE, LINE)
  - A document containing this term is more likely to be relevant than a document that doesn't, but it's not a sure indicator of relevance.
  - → For frequent terms, we want positive weights for words like HIGH, INCREASE, and LINE, but lower weights than for rare terms.

- 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.
idf weight

- $df_t$ is the document frequency, the number of documents that $t$ occurs in.
idf weight

- $df_t$ is the document frequency, the number of documents that $t$ occurs in.
- $df$ is an inverse measure of the informativeness of the term.
idf weight

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

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

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

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

- idf is a measure of the informativeness of the term.
idf weight

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

$$\text{idf}_t = \log_{10} \frac{N}{df_t}$$

- idf is a measure of the informativeness of the term.
- We use $\log N/df_t$ instead of $N/df_t$ to “dampen” the effect of idf.
idf weight

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

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

- idf is a measure of the informativeness of the term.
- We use $\log N/df_t$ instead of $N/df_t$ to “dampen” the effect of idf.
- So we use the log transformation for both term frequency and document frequency.
Examples for idf

Compute $\text{idf}_t$ using the formula: $\text{idf}_t = \log_{10} \left( \frac{1,000,000}{\text{df}_t} \right)$

<table>
<thead>
<tr>
<th>term</th>
<th>$\text{df}_t$</th>
<th>$\text{idf}_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>calpurnia</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>sunday</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td></td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td></td>
</tr>
</tbody>
</table>
Examples for idf

Compute $idf_t$ using the formula: $idf_t = \log_{10} \frac{1,000,000}{df_t}$

<table>
<thead>
<tr>
<th>term</th>
<th>$df_t$</th>
<th>$idf_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>calpurnia</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>sunday</td>
<td>1000</td>
<td>3</td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td>2</td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td>0</td>
</tr>
</tbody>
</table>
Effect of idf on ranking

- idf affects the ranking of documents only if the query has at least two terms.
Effect of idf on ranking

- idf affects the ranking of documents only if the query has 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.
Effect of idf on ranking

- idf affects the ranking of documents only if the query has 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.
- idf has no effect on ranking for one-term queries.
Effect of idf on ranking

- idf affects the ranking of documents only if the query has 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.
- idf has no effect on ranking for one-term queries.
- Questions about idf?
## Collection frequency vs. Document frequency

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSURANCE</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>TRY</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- The collection frequency of $t$ is the number of tokens of $t$ in the collection where we count multiple occurrences.
Collection frequency vs. Document frequency

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSURANCE</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>TRY</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- The collection frequency of $t$ is the number of tokens of $t$ in the collection where we count multiple occurrences.

- Why these numbers?
### Collection frequency vs. Document frequency

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSURANCE</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>TRY</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- The collection frequency of $t$ is the number of tokens of $t$ in the collection where we count multiple occurrences.
- **Why these numbers?**
- **Which word is a better search term (and should get a higher weight)?**
Recap  Term frequency  tf-idf weighting  The vector space

## Collection frequency vs. Document frequency

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSURANCE</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>TRY</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- The collection frequency of $t$ is the number of tokens of $t$ in the collection where we count multiple occurrences.
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df is better for weighting than cf.
The tf-idf weight of a term is the product of its tf weight and its idf weight.
The tf-idf weight of a term is the product of its tf weight and its idf weight.

\[ w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t} \]
The tf-idf weight of a term is the product of its tf weight and its idf weight.

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

Best known weighting scheme in information retrieval.
The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

Best known weighting scheme in information retrieval

Note: the “-” in tf-idf is a hyphen, not a minus sign!
The tf-idf weight of a term is the **product of its tf weight and its idf weight**.

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- 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
Summary: tf-idf

- Assign a tf-idf weight for each term $t$ in each document $d$:
  $$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$
Summary: tf-idf

- Assign a tf-idf weight for each term $t$ in each document $d$:
  $$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$
- $N$: total number of documents
Summary: tf-idf

- Assign a tf-idf weight for each term $t$ in each document $d$:
  \[ w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t} \]
- $N$: total number of documents
- Increases with the number of occurrences within a document
Summary: tf-idf

- Assign a tf-idf weight for each term $t$ in each document $d$:
  \[ w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{df_t} \]

- $N$: total number of documents
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection
## Term, collection and document frequency

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>term frequency</td>
<td>$tf_{t,d}$</td>
<td>number of occurrences of $t$ in $d$</td>
</tr>
<tr>
<td>document frequency</td>
<td>$df_t$</td>
<td>number of documents in the collection that $t$ occurs in</td>
</tr>
<tr>
<td>collection frequency</td>
<td>$cf_t$</td>
<td>total number of occurrences of $t$ in the collection</td>
</tr>
</tbody>
</table>
## Term, collection and document frequency

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>term frequency</td>
<td>$tf_{t,d}$</td>
<td>number of occurrences of $t$ in $d$</td>
</tr>
<tr>
<td>document frequency</td>
<td>$df_t$</td>
<td>number of documents in the collection that $t$ occurs in</td>
</tr>
<tr>
<td>collection frequency</td>
<td>$cf_t$</td>
<td>total number of occurrences of $t$ in the collection</td>
</tr>
</tbody>
</table>
## Term, collection and document frequency

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>term frequency</td>
<td>$tf_{t,d}$</td>
<td>number of occurrences of $t$ in $d$</td>
</tr>
<tr>
<td>document frequency</td>
<td>$df_t$</td>
<td>number of documents in the collection that $t$ occurs in</td>
</tr>
<tr>
<td>collection frequency</td>
<td>$cf_t$</td>
<td>total number of occurrences of $t$ in the collection</td>
</tr>
</tbody>
</table>
Outline

1 Recap

2 Term frequency

3 tf-idf weighting

4 The vector space
Recap Term frequency tf-idf weighting The vector space

Binary → count → weight matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anthony</strong></td>
<td>5.25</td>
<td>3.18</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td><strong>Brutus</strong></td>
<td>1.21</td>
<td>6.10</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Caesar</strong></td>
<td>8.59</td>
<td>2.54</td>
<td>0.0</td>
<td>1.51</td>
<td>0.25</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Calpurnia</strong></td>
<td>0.0</td>
<td>1.54</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Cleopatra</strong></td>
<td>2.85</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Mercy</strong></td>
<td>1.51</td>
<td>0.0</td>
<td>1.90</td>
<td>0.12</td>
<td>5.25</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td><strong>Worser</strong></td>
<td>1.37</td>
<td>0.0</td>
<td>0.11</td>
<td>4.15</td>
<td>0.25</td>
<td>1.95</td>
<td></td>
</tr>
</tbody>
</table>

...  

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$. 
### Recap: Term frequency → tf-idf weighting → The vector space

#### Binary → count → weight matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ANTHONY</strong></td>
<td>5.25</td>
<td>3.18</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td><strong>BRUTUS</strong></td>
<td>1.21</td>
<td>6.10</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>CAESAR</strong></td>
<td>8.59</td>
<td>2.54</td>
<td>0.0</td>
<td>1.51</td>
<td>0.25</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>CALPURNIA</strong></td>
<td>0.0</td>
<td>1.54</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>CLEOPATRA</strong></td>
<td>2.85</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td><strong>MERCY</strong></td>
<td>1.51</td>
<td>0.0</td>
<td>1.90</td>
<td>0.12</td>
<td>5.25</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td><strong>WORSER</strong></td>
<td>1.37</td>
<td>0.0</td>
<td>0.11</td>
<td>4.15</td>
<td>0.25</td>
<td>1.95</td>
<td></td>
</tr>
</tbody>
</table>

Each document is now represented by a **real-valued vector** of tf-idf weights \( \in \mathbb{R}^{|V|} \).
Documents as vectors

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

So we have a $|V|$-dimensional real-valued vector space.
Documents as vectors

- Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are *axes* of the space.
Documents as vectors

- Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
Documents as vectors

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

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 a web search engine.

This is a very sparse vector - most entries are zero.
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query
- proximity = similarity
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query
- proximity \approx \text{negative distance}
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query
  - proximity = similarity
  - proximity ≈ negative distance
- Recall: We’re doing this because we want to get away from the you’re-either-in-or-out Boolean model.
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query
  - proximity = similarity
  - proximity \approx \text{negative distance}
- Recall: We’re doing this because we want to get away from the you’re-either-in-or-out Boolean model.
- Instead: rank more relevant documents higher than less relevant documents
How do we formalize vector space similarity?

- First cut: distance between two points
How do we formalize vector space similarity?

- First cut: distance between two points
- \( = \text{distance between the end points of the two vectors} \)
How do we formalize vector space similarity?

- First cut: distance between two points
- \( = \) distance between the end points of the two vectors
- Euclidean distance?
How do we formalize vector space similarity?

- First cut: distance between two points
- ( = distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
How do we formalize vector space similarity?

- First cut: distance between two points
- \( = \) distance between the end points of the two vectors
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.
Why distance is a bad idea
Why distance is a bad idea

The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query $q$ and the distribution of terms in the document $d_2$ are very similar.
Why distance is a bad idea

The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query $q$ and the distribution of terms in the document $d_2$ are very similar.

Questions about basic vector space setup?
Use angle instead of distance

- Rank documents according to angle with query
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 $d'$. 
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 $d'$.
- “Semantically” $d$ and $d'$ have the same content.
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 $d'$.
- “Semantically” $d$ and $d'$ have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity.
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 $d'$.
- “Semantically” $d$ and $d'$ have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity.
- The Euclidean distance between the two documents can be quite large.
From angles to cosines

- The following two notions are equivalent.
From angles to cosines

- The following two notions are equivalent.
  - Rank documents according to the angle between query and document in decreasing order.
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
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 $\cos \theta(query, document)$ in increasing order
- Cosine is a monotonically decreasing function of the angle for the interval $[0^\circ, 180^\circ]$
Cosine
What about angles $> 180^\circ$?
Length normalization

- How do we compute the cosine?
How do we compute the cosine?

A vector can be (length-) normalized by dividing each of its components by its length – here we use the $L_2$ norm:

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

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the $L_2$ norm:
  \[ ||x||_2 = \sqrt{\sum_i x_i^2} \]
- This maps vectors onto the unit sphere . . .
Length normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the $L_2$ norm:
  \[ \|x\|_2 = \sqrt{\sum_i x_i^2} \]
- This maps vectors onto the unit sphere . . .
- . . . since after normalization: \[ \|x\|_2 = \sqrt{\sum_i x_i^2} = 1.0 \]
Length normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the $L_2$ norm:
  $$||x||_2 = \sqrt{\sum_i x_i^2}$$
- This maps vectors onto the unit sphere . . .
- . . . since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- As a result, longer documents and shorter documents have weights of the same order of magnitude.
Length normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the $L_2$ norm:
  \[ ||x||_2 = \sqrt{\sum_i x_i^2} \]
- This maps vectors onto the unit sphere ...
- ... since after normalization: \[ ||x||_2 = \sqrt{\sum_i x_i^2} = 1.0 \]
- As a result, longer documents and shorter documents have weights of the same order of magnitude.
- Effect on the two documents $d$ and $d'$ ($d$ appended to itself) from earlier slide: they have identical vectors after length-normalization.
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.
Cosine similarity between query and document

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

- $q_i$ is the tf-idf weight of term $i$ in the query.
- $d_i$ is the tf-idf weight of term $i$ in the document.
Cosine similarity between query and document

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

- \(q_i\) 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 \(\vec{d}\).
Cosine similarity between query and document

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

- \(q_i\) is the tf-idf weight of term \(i\) in the query.
- \(d_i\) is the tf-idf weight of term \(i\) in the document.
- \(|\mathbf{q}|\) and \(|\mathbf{d}|\) are the lengths of \(\mathbf{q}\) and \(\mathbf{d}\).
- This is the cosine similarity of \(\mathbf{q}\) and \(\mathbf{d}\) or, equivalently, the cosine of the angle between \(\mathbf{q}\) and \(\mathbf{d}\).
Cosine for normalized vectors

For normalized vectors, the cosine is equivalent to the dot product or scalar product.
Cosine for normalized vectors

- For normalized vectors, the cosine is equivalent to the dot product or scalar product.
- \( \cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i \) (if \( \vec{q} \) and \( \vec{d} \) are length-normalized).
Cosine similarity illustrated
Cosine: Example

How similar are the novels? SaS: Sense and Sensibility, PaP: Pride and Prejudice, and WH: Wuthering Heights?
**Cosine: Example**

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

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
</tbody>
</table>
Cosine: Example

term frequencies (counts)

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
</tbody>
</table>
### Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>
### Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

(To simplify this example, we don’t do idf weighting.)
### Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>
Recap Term frequency tf-idf weighting

The vector space

Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

log frequency weighting

& cosine normalization

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>0.335</td>
<td>0.0</td>
<td>0.405</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0.0</td>
<td>0.0</td>
<td>0.588</td>
</tr>
</tbody>
</table>
Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0.0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0.0</td>
<td>0.0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>0.335</td>
<td>0.0</td>
<td>0.405</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0.0</td>
<td>0.0</td>
<td>0.588</td>
</tr>
</tbody>
</table>

\[ \cos(SaS, PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94. \]
### Cosine: Example

#### Log frequency weighting

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

#### Log frequency weighting & cosine normalization

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>0.335</td>
<td>0.0</td>
<td>0.405</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0.0</td>
<td>0.0</td>
<td>0.588</td>
</tr>
</tbody>
</table>

- $\cos(SaS, PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94.$
- $\cos(SaS, WH) \approx 0.79.$
### Cosine: Example

**log frequency weighting**

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

**log frequency weighting & cosine normalization**

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>0.789</td>
<td>0.832</td>
<td>0.524</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>0.515</td>
<td>0.555</td>
<td>0.465</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>0.335</td>
<td>0.0</td>
<td>0.405</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0.0</td>
<td>0.0</td>
<td>0.588</td>
</tr>
</tbody>
</table>

\[
\cos(SaS, PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94. \\
\cos(SaS, WH) \approx 0.79 \\
\cos(PaP, WH) \approx 0.69
\]
### Cosine: Example

<table>
<thead>
<tr>
<th>term</th>
<th>SaS</th>
<th>PaP</th>
<th>WH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFECTION</td>
<td>3.06</td>
<td>2.76</td>
<td>2.30</td>
</tr>
<tr>
<td>JEALOUS</td>
<td>2.0</td>
<td>1.85</td>
<td>2.04</td>
</tr>
<tr>
<td>GOSSIP</td>
<td>1.30</td>
<td>0</td>
<td>1.78</td>
</tr>
<tr>
<td>WUTHERING</td>
<td>0</td>
<td>0</td>
<td>2.58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>log frequency weighting &amp; cosine normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>AFFECTION</td>
</tr>
<tr>
<td>JEALOUS</td>
</tr>
<tr>
<td>GOSSIP</td>
</tr>
<tr>
<td>WUTHERING</td>
</tr>
</tbody>
</table>

- \( \cos(SaS,PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94 \).
- \( \cos(SaS,WH) \approx 0.79 \).
- \( \cos(PaP,WH) \approx 0.69 \).
- **Why do we have \( \cos(SaS,PaP) > \cos(SaS,WH) \)?**
Computing the cosine score

**CosineScore(q)**

1. `float Scores[N] = 0`
2. `float Length[N]`
3. **for each** query term `t`
4. **do** calculate `wt,q` and fetch postings list for `t`
5. **for each** pair(`d, tf_t,d`) in postings list
6. **do** `Scores[d] + = wt,d × wt,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[]`
## Components of tf-idf weighting

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

Schütze: Scoring, term weighting, the vector space model 49 / 53
## Components of tf-idf weighting

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>n (natural)</td>
<td>tf&lt;sub&gt;t,d&lt;/sub&gt;</td>
<td>n (no) 1</td>
</tr>
<tr>
<td>l (logarithm)</td>
<td>1 + log(tf&lt;sub&gt;t,d&lt;/sub&gt;)</td>
<td>t (idf) log ( \frac{N}{df_t} )</td>
</tr>
<tr>
<td>a (augmented)</td>
<td>0.5 + ( \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})} )</td>
<td>p (prob idf) ( \max{0, \log \frac{N - df_t}{df_t}} )</td>
</tr>
<tr>
<td>b (boolean)</td>
<td>( \begin{cases} 1 &amp; \text{if } tf_{t,d} &gt; 0 \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>( \text{b (byte size)} ) ( \frac{1}{\text{CharLength}^\alpha}, \alpha &lt; 1 )</td>
</tr>
<tr>
<td>L (log ave)</td>
<td>( \frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}<em>{t \in d}(tf</em>{t,d}))} )</td>
<td>( \text{b (byte size)} ) ( \frac{1}{\text{CharLength}^\alpha}, \alpha &lt; 1 )</td>
</tr>
</tbody>
</table>

Best known combination of weighting options
## Components of tf-idf weighting

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>n (natural)</td>
<td>tf&lt;sub&gt;t,d&lt;/sub&gt;</td>
<td>n (no) 1</td>
</tr>
<tr>
<td>l (logarithm)</td>
<td>1 + log(tf&lt;sub&gt;t,d&lt;/sub&gt;)</td>
<td>t (idf) log ( \frac{N}{df_t} )</td>
</tr>
<tr>
<td>a (augmented)</td>
<td>0.5 + ( \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})} )</td>
<td>p (prob idf) max{0, log ( \frac{N-df_t}{df_t} ) }</td>
</tr>
<tr>
<td>b (boolean)</td>
<td>( \begin{cases} 1 &amp; \text{if } tf_{t,d} &gt; 0 \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>u (pivoted unique) 1 ( \div ) u</td>
</tr>
<tr>
<td>L (log ave)</td>
<td>( \frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}<em>t∈d(tf</em>{t,d}))} )</td>
<td>b (byte size) ( 1/\text{CharLength}^{\alpha} ), ( \alpha &lt; 1 )</td>
</tr>
</tbody>
</table>

Default: no weighting
tf-idf example

- We often use **different weightings** for queries and documents.
tf-idf example

- We often use different weightings for queries and documents.
- Notation: qqq.ddd
tf-idf example

- We often use **different weightings** for queries and documents.
- Notation: qqq.ddd
- Example: ltn.lnc
We often use different weightings for queries and documents.

Notation: qqq.ddd

Example: ltn.lnc

query: logarithmic tf, idf, no normalization
We often use **different weightings** for queries and documents.

- Notation: qqq.ddd
- Example: ltn.lnc

query: logarithmic tf, idf, no normalization

document: logarithmic tf, no df weighting, cosine normalization
tf-idf example

- We often use different weightings for queries and documents.
- Notation: qqq.ddd
- Example: ltn. Inc
- query: logarithmic tf, idf, no normalization
- document: logarithmic tf, no df weighting, cosine normalization
- Isn’t it bad to not idf-weight the document?
tf-idf example

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

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

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
</tr>
<tr>
<td>auto</td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
## tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>document</th>
<th></th>
<th></th>
<th></th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>df</td>
<td>idf</td>
<td>weight</td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>weight</td>
<td>n’lized</td>
<td></td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th>document</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>df</td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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.
### tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th>document</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>df</td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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.
### tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th></th>
<th></th>
<th>document</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>df</td>
<td>idf</td>
<td>weight</td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>weight</td>
<td>n’lized</td>
<td>product</td>
<td></td>
<td></td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
Recap Term frequency tf-idf weighting

The vector space

## tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td></td>
<td></td>
<td>2</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
## tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th>document</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
<td>df</td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
</tr>
</tbody>
</table>

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.
**tf-idf example: ltn.lnc**

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

<table>
<thead>
<tr>
<th>word</th>
<th>query tf-raw</th>
<th>query tf-wght</th>
<th>query df</th>
<th>query idf</th>
<th>query weight</th>
<th>document tf-raw</th>
<th>document tf-wght</th>
<th>document weight</th>
<th>document n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
**tf-idf example: ltn.lnc**

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td>1.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
**tf-idf example: ltn.lnc**

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

<table>
<thead>
<tr>
<th>word</th>
<th>query</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tf-raw</td>
<td>tf-wght</td>
</tr>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

\[
\frac{1}{1.92} \approx 0.52
\]

\[
\frac{1.3}{1.92} \approx 0.68
\]
## tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>1.04</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td>1.3</td>
<td>0.68</td>
<td>2.04</td>
</tr>
</tbody>
</table>

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.
tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>1.04</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td>1.3</td>
<td>0.68</td>
<td>2.04</td>
</tr>
</tbody>
</table>

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.

Final similarity score between query and document: \( \sum_i w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08 \)
### tf-idf example: ltn.lnc

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

<table>
<thead>
<tr>
<th>word</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>df</th>
<th>idf</th>
<th>weight</th>
<th>query</th>
<th>tf-raw</th>
<th>tf-wght</th>
<th>weight</th>
<th>n’lized</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>2.3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>0</td>
</tr>
<tr>
<td>best</td>
<td>1</td>
<td>1</td>
<td>50000</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>10000</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.52</td>
<td>1.04</td>
</tr>
<tr>
<td>insurance</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td>3.0</td>
<td>3.0</td>
<td>2</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>0.68</td>
<td>2.04</td>
</tr>
</tbody>
</table>

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

Final similarity score between query and document: $\sum_i w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08$

Questions?
Summary: Ranked retrieval in the vector space model

- Represent the query as a weighted tf-idf vector
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
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
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
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
Recap  Term frequency  tf-idf weighting  The vector space

Resources

- Chapters 6 and 7 of IIR
Resources

- Chapters 6 and 7 of IIR
- Resources at http://ifnlp.org/ir
Resources

- Chapters 6 and 7 of IIR
- Resources at http://ifnlp.org/ir
- Vector space for dummies
Resources

- Chapters 6 and 7 of IIR
- Resources at http://ifnlp.org/ir
- Vector space for dummies
- Exploring the similarity space (Moffat and Zobel, 2005)
Resources

- Chapters 6 and 7 of IIR
- Resources at http://ifnlp.org/ir
- Vector space for dummies
- Exploring the similarity space (Moffat and Zobel, 2005)
- Okapi BM25 (a state-of-the-art weighting method, 11.4.3 of IIR)