Part I: Web Structure Mining
Chapter 1: Information Retrieval and Web Search

• The Web Challenges
• Crawling the Web
• Indexing and Keyword Search
• Evaluating Search Quality
• Similarity Search
The Web Challenges


Slides for Chapter 1: Information Retrieval and Web Search
The Web Challenges

18 years later …

• The recent Web is huge and grows incredibly fast. About ten years after the Tim Berners-Lee proposal the Web was estimated to 150 million nodes (pages) and 1.7 billion edges (links). Now it includes more than 4 billion pages, with about a million added every day.

• Restricted formal semantics - nodes are just web pages and links are of a single type (e.g. “refer to”). The meaning of the nodes and links is not a part of the web system, rather it is left to the web page developers to describe in the page content what their web documents mean and what kind of relations they have with the documented they link to.

• As there is no central authority or editors relevance, popularity or authority of web pages are hard to evaluate. Links are also very diverse and many have nothing to do with content or authority (e.g. navigation links).
The Web Challenges

How to turn the web data into web knowledge

• Use the existing Web
  – Web Search Engines
  – Topic Directories
• Change the Web
  – Semantic Web
Crawling The Web

- To make Web search efficient search engines collect web documents and index them by the words (terms) they contain.
- For the purposes of indexing web pages are first collected and stored in a local repository
- Web crawlers (also called spiders or robots) are programs that systematically and exhaustively browse the Web and store all visited pages
- Crawlers follow the hyperlinks in the Web documents implementing graph search algorithms like depth-first and breadth-first
Crawling The Web

Depth-first Web crawling limited to depth 3


Slides for Chapter 1: Information Retrieval and Web Search
Crawling The Web

Breadth-first Web crawling limited to depth 3

Slides for Chapter 1: Information Retrieval an Web Search
Crawling The Web

Issues in Web Crawling:

• Network latency (multithreading)
• Address resolution (DNS caching)
• Extracting URLs (use canonical form)
• Managing a huge web page repository
• Updating indices
• Responding to constantly changing Web
• Interaction of Web page developers
• Advanced crawling by guided (informed) search (using web page ranks)
We need efficient content-based access to Web documents

- **Document representation:**
  - Term-document matrix (inverted index)

- **Relevance ranking:**
  - Vector space model
Indexing and Keyword Search

Creating term-document matrix (inverted index)

- Documents are tokenized (punctuation marks are removed and the character strings without spaces are considered as tokens)
- All characters are converted to upper or to lower case.
- Words are reduced to their canonical form (*stemming*)
- *Stopwords* (*a, an, the, on, in, at*, etc.) are removed.

The remaining words, now called *terms* are used as *features* (*attributes*) in the term-document matrix
### CCSU Departments example

#### Document statistics

<table>
<thead>
<tr>
<th>Document ID</th>
<th>Document name</th>
<th>Words</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_1</td>
<td>Anthropology</td>
<td>114</td>
<td>86</td>
</tr>
<tr>
<td>d_2</td>
<td>Art</td>
<td>153</td>
<td>105</td>
</tr>
<tr>
<td>d_3</td>
<td>Biology</td>
<td>123</td>
<td>91</td>
</tr>
<tr>
<td>d_4</td>
<td>Chemistry</td>
<td>87</td>
<td>58</td>
</tr>
<tr>
<td>d_5</td>
<td>Communication</td>
<td>124</td>
<td>88</td>
</tr>
<tr>
<td>d_6</td>
<td>Computer Science</td>
<td>101</td>
<td>77</td>
</tr>
<tr>
<td>d_7</td>
<td>Criminal Justice</td>
<td>85</td>
<td>60</td>
</tr>
<tr>
<td>d_8</td>
<td>Economics</td>
<td>107</td>
<td>76</td>
</tr>
<tr>
<td>d_9</td>
<td>English</td>
<td>116</td>
<td>80</td>
</tr>
<tr>
<td>d_10</td>
<td>Geography</td>
<td>95</td>
<td>68</td>
</tr>
<tr>
<td>d_11</td>
<td>History</td>
<td>108</td>
<td>78</td>
</tr>
<tr>
<td>d_12</td>
<td>Mathematics</td>
<td>89</td>
<td>66</td>
</tr>
<tr>
<td>d_13</td>
<td>Modern Languages</td>
<td>110</td>
<td>75</td>
</tr>
<tr>
<td>d_14</td>
<td>Music</td>
<td>137</td>
<td>91</td>
</tr>
<tr>
<td>d_15</td>
<td>Philosophy</td>
<td>85</td>
<td>54</td>
</tr>
<tr>
<td>d_16</td>
<td>Physics</td>
<td>130</td>
<td>100</td>
</tr>
<tr>
<td>d_17</td>
<td>Political Science</td>
<td>120</td>
<td>86</td>
</tr>
<tr>
<td>d_18</td>
<td>Psychology</td>
<td>96</td>
<td>60</td>
</tr>
<tr>
<td>d_19</td>
<td>Sociology</td>
<td>99</td>
<td>66</td>
</tr>
<tr>
<td>d_20</td>
<td>Theatre</td>
<td>116</td>
<td>80</td>
</tr>
</tbody>
</table>

**Total number of words/terms**: 2195  1545  
**Number of different words/terms**: 744  671
### CCSU Departments example

**Boolean (Binary) Term Document Matrix**

<table>
<thead>
<tr>
<th>DID</th>
<th>lab</th>
<th>laboratory</th>
<th>programming</th>
<th>computer</th>
<th>program</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₃</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₆</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d₇</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₈</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₉</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d₁₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₄</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d₁₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₁₆</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₁₇</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₁₈</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₉</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d₂₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


Slides for Chapter 1: Information Retrieval an Web Search
CCSU Departments example

Term document matrix with positions

<table>
<thead>
<tr>
<th>DID</th>
<th>lab</th>
<th>laboratory</th>
<th>programming</th>
<th>computer</th>
<th>program</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[71]</td>
</tr>
<tr>
<td>d₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[7]</td>
</tr>
<tr>
<td>d₃</td>
<td>0</td>
<td>[65,69]</td>
<td>0</td>
<td>[68]</td>
<td>0</td>
</tr>
<tr>
<td>d₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[26]</td>
<td>[30,43]</td>
</tr>
<tr>
<td>d₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₆</td>
<td>0</td>
<td>0</td>
<td>[40,42]</td>
<td>[1,3,7,13,26,34]</td>
<td>[11,18,61]</td>
</tr>
<tr>
<td>d₇</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[9,42]</td>
</tr>
<tr>
<td>d₈</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[57]</td>
</tr>
<tr>
<td>d₉</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₁</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[17]</td>
<td>0</td>
</tr>
<tr>
<td>d₁₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₄</td>
<td>[42]</td>
<td>0</td>
<td>0</td>
<td>[41]</td>
<td>[71]</td>
</tr>
<tr>
<td>d₁₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[37,38]</td>
</tr>
<tr>
<td>d₁₆</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[81]</td>
</tr>
<tr>
<td>d₁₇</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[68]</td>
</tr>
<tr>
<td>d₁₈</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d₁₉</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[51]</td>
</tr>
<tr>
<td>d₂₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Slides for Chapter 1: Information Retrieval an Web Search
Vector Space Model

Boolean representation

- documents $d_1, d_2, \ldots, d_n$
- terms $t_1, t_2, \ldots, t_m$
- term $t_i$ occurs $n_{ij}$ times in document $d_j$. 
- **Boolean representation:**

\[
\tilde{d}_j = (d^1_j \ d^2_j \ldots \ d^m_j) \\
\]

\[
d^i_j = \begin{cases} 
0 & \text{if } n_{ij} = 0 \\
1 & \text{if } n_{ij} > 0 
\end{cases}
\]

- For example, if the terms are: *lab, laboratory, programming, computer* and *program*. Then the Computer Science document is represented by the Boolean vector

\[
\tilde{d}_6 = (0 \ 0 \ 1 \ 1 \ 1)
\]
Term Frequency (TF) representation

Document vector \( \vec{d}_j = (d^1_j, d^2_j, \ldots, d^m_j) \) with components \( d^i_j = TF(t_i, d_j) \)

- Using the sum of term counts: \( TF(t_i, d_j) = \frac{n_{ij}}{\sum_{k=1}^{m} n_{kj}} \) if \( n_{ji} > 0 \)

- Using the maximum of term counts: \( TF(t_i, d_j) = \max_k \frac{n_{ij}}{n_{kj}} \) if \( n_{ji} > 0 \)

- Cornell SMART system: \( TF(t_i, d_j) = \frac{1}{\log(1 + \log n_{ji})} \) if \( n_{ji} > 0 \)
Inverted Document Frequency (IDF)

Document collection: $D = \bigcup_{1}^{n} d_j$, documents that contain term $t_i$: $D_{ti} = \{ d_j \mid n_{ij} > 0 \}$

- Simple fraction: $IDF(t_i) = \frac{|D|}{|D_{ti}|}$

  or

- Using a log function: $IDF(t_i) = \log \frac{1 + |D|}{|D_{ti}|}$
TFIDF representation

\[ d_j^i = TF(t_i, d_j) \times IDF(t_i) \]

For example, the \textit{computer science} TF vector

\[ \vec{d}_6 = (0 \ 0 \ 0.026 \ 0.078 \ 0.039) \]

scaled with the IDF of the terms

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
lab & laboratory & Programming & computer & program \\
\hline
3.04452 & 3.04452 & 3.04452 & 1.43508 & 0.559616 \\
\hline
\end{tabular}
\end{center}

results in

\[ \vec{d}_6 = (0 \ 0 \ 0.079 \ 0.112 \ 0.022) \]
Relevance Ranking

- Represent the query as a vector \( q = \{ \text{computer, program} \} \)
  \( \vec{q} = (0 \ 0 \ 0 \ 0.5 \ 0.5) \)

- Apply IDF to its components

<table>
<thead>
<tr>
<th></th>
<th>lab</th>
<th>laboratory</th>
<th>Programming</th>
<th>computer</th>
<th>program</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.04452</td>
<td>3.04452</td>
<td>3.04452</td>
<td>1.43508</td>
<td>0.559616</td>
</tr>
</tbody>
</table>

\( \vec{q} = (0 \ 0 \ 0 \ 0.718 \ 0.28) \)

- Use Euclidean norm of the vector difference
  \( \| \vec{q} - \vec{d}_j \| = \sqrt{\sum_{i=1}^{m} (q^i - d^i_j)^2} \)

- or Cosine similarity (equivalent to dot product for normalized vectors)
  \( \vec{q} \cdot \vec{d}_j = \sum_{i=1}^{m} q^i d^i_j \)

Slides for Chapter 1: Information Retrieval and Web Search
## Relevance Ranking

Cosine similarities and distances to $\tilde{q} = (0 \ 0 \ 0 \ 0.932 \ 0.363)$ (normalized)

| Doc | TFIDF Coordinates (normalized) | $\tilde{q}.\tilde{d}_j$ (rank) | $|\tilde{q} - \tilde{d}_j|$ (rank) |
|-----|--------------------------------|-------------------------------|---------------------------------|
| $d_1$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_2$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_3$ | 0 0.972 0 0.234 0 | 0.218 | 1.250 |
| $d_4$ | 0 0 0 0.783 0.622 | 0.956 (1) | 0.298 (1) |
| $d_5$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_6$ | 0 0 0.559 0.811 0.172 | 0.819 (2) | 0.603 (2) |
| $d_7$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_8$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_9$ | 0 0 0 0 0 | 0 | 1 |
| $d_{10}$ | 0 0 0 0 0 | 0 | 1 |
| $d_{11}$ | 0 0 0 0 0 | 0 | 1 |
| $d_{12}$ | 0 0 0 0 1 | 0.932 | 0.369 |
| $d_{13}$ | 0 0 0 0 0 | 0 | 1 |
| $d_{14}$ | 0.890 0 0 0.424 0.167 | 0.456 (3) | 1.043 (3) |
| $d_{15}$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_{16}$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_{17}$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $d_{18}$ | 0 0 0 0 0 | 0 | 1 |
| $d_{19}$ | 0 0 0 0 1 | 0.363 | 1.129 |
| $D_{20}$ | 0 0 0 0 0 | 0 | 1 |

Slides for Chapter 1: Information Retrieval an Web Search
Relevance Feedback

- The user provides feedback:
  - Relevant documents $D_+$
  - Irrelevant documents $D_-$

- The original query vector $\vec{q}$ is updated (Rocchio’s method)

\[
\vec{q}' = \alpha \vec{q} + \beta \sum_{d_j \in D_+} \vec{d}_j - \gamma \sum_{d_j \in D_-} \vec{d}_j
\]

- Pseudo-relevance feedback
  - Top 10 documents returned by the original query belong to $D_+$
  - The rest of documents belong to $D_-$
Advanced text search

- Using "OR" or "NOT" boolean operators

- Phrase Search
  - Statistical methods to extract phrases from text
  - Indexing phrases

- Part-of-speech tagging

- Approximate string matching (using n-grams)
  - Example: match "program" and "prorgam"
    $$\{pr, ro, og, gr, ra, am\} \cap \{pr, ro, or, rg, ga, am\} = \{pr, ro, am\}$$
Using the HTML structure in keyword search

• Titles and metatags
  – Use them as tags in indexing
  – Modify ranking depending on the context where the term occurs

• Headings and font modifiers (prone to spam)

• Anchor text
  – Plays an important role in web page indexing and search
  – Allows to increase search indices with pages that have never been crawled
  – Allows to index non-textual content (such as images and programs)
Evaluating search quality

• Assume that there is a set of queries $Q$ and a set of documents $D$, and for each query $q \in Q$ submitted to the system we have:
  – The response set of documents (retrieved documents) $R_q \subseteq D$
  – The set of relevant documents $D_q$ selected manually from the whole collection of documents, i.e. $D_q \subseteq D$

\[
\text{precision} = \frac{|D_q \cap R_q|}{|R_q|} \\
\text{recall} = \frac{|D_q \cap R_q|}{|D_q|}
\]
Precision-recall framework (set-valued)

- Determine the relationship between the set of relevant documents ($D_q$) and the set of retrieved documents ($R_q$)

- Ideally $D_q \supseteq R_q$

- Generally $D_q \cap R_q \subset D_q$

- A very general query leads to recall $= 1$, but low precision

- A very restrictive query leads to precision $= 1$, but low recall

- A good balance is needed to maximize both precision and recall
Precision-recall framework (using ranks)

- With thousands of documents finding $D_q$ is practically impossible.
- So, let’s consider a list $R_q = (d_1, d_2, ..., d_m)$ of ranked documents (highest rank first)
- For each $d_i \in R_q$ compute its relevance as $r_i = \begin{cases} 1 & \text{if } d_i \in D_q \\ 0 & \text{otherwise} \end{cases}$

- Then define precision at rank $k$ as
  \[\text{precision} (k) = \frac{1}{k} \sum_{i=1}^{k} r_i\]

- And recall at rank $k$ as
  \[\text{recall} (k) = \frac{1}{|D_q|} \sum_{i=1}^{k} r_i\]
## Precision-recall framework (example)

Relevant documents \( D_q = (d_4, d_6, d_{14}) \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>Document index</th>
<th>( r_k )</th>
<th>( \text{recall} \ (k) )</th>
<th>( \text{precision} \ (k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0.333</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>0</td>
<td>0.333</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0.667</td>
<td>0.667</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.429</td>
</tr>
</tbody>
</table>

Slides for Chapter 1: Information Retrieval an Web Search
Precision-recall framework

Average precision = \frac{1}{|D_q|} \sum_{k=1}^{\left|D\right|} r_k \times \text{precision}(k)

- Combines precision and recall and also evaluates document ranking
- The maximal value of 1 is reached when all relevant documents are retrieved and ranked before any irrelevant ones.
- Practically to compute the average precision we first go over the ranked documents from $R_q$ and then continue with the rest of the documents from $D$ until all relevant documents are included.
Similarity Search

• *Cluster hypothesis* in IR: Documents similar to relevant documents are likely to be relevant too

• Once a relevant document is found a larger collection of possibly relevant documents may be found by retrieving similar documents.

• Similarity measure between documents
  – Cosine Similarity
  – Jaccard Similarity (most popular approach)
  – Document Resemblance

Slides for Chapter 1: Information Retrieval an Web Search
Cosine Similarity

• Given document $d$ and collection $D$ the problem is to find a number (usually 10 or 20) of documents $d_i \in D$, which have the largest value of cosine similarity to $d$.

• No problems with the small query vector, so we can use more (or all) dimensions of the vector space

• How many and which terms to use?
  o All terms from the corpus
  o Select the terms that best represent the documents (*Feature Selection*)
    – Use highest TF score
    – Use highest IDF score
    – Combine TF and IDF scores (e.g. TFIDF)
Jaccard Similarity

• Use Boolean document representation and only the nonzero coordinates of the vectors (i.e. those that are 1)

• Jaccard Coefficient: proportion of coordinates that are 1 in both documents to those that are 1 in either of the documents

\[
sim(\vec{d}_1, \vec{d}_2) = \frac{\left\{ j \mid d_1^j = 1 \land d_2^j = 1 \right\}} {\left\{ j \mid d_1^j = 1 \lor d_2^j = 1 \right\}}
\]

• Set formulation \( \sim(d_1, d_2) = \frac{|T(d_1) \cap T(d_2)|}{|T(d_1) \cup T(d_2)|} \)
Computing Jaccard Coefficient

• Problems
  – Direct computation is straightforward, but with large collections it may lead to inefficient similarity search.
  – Finding similar documents at query time is impractical.

• Solutions
  – Do most of the computation offline
  – Create a list of all document pairs sorted by the similarity of the documents in each pair. Then the $k$ most similar documents to a given document $d$ are those that are paired with $d$ in the first $k$ pairs from the list.
  – Eliminate frequent terms
  – Pair documents that share at least one term
Document Resemblance

• The task is finding identical or nearly identical documents, or documents that share phrases, sentences or paragraphs.

• The set of words approach does not work.

• Consider the document as a sequence of words and extract from this sequence short subsequences with fixed length (n-grams or shingles).

• Represent document $d$ as a set of $w$-grams $S(d,w)$

• Example: $T(d) = \{a, b, c, d, e\}$ $S(d,2) = \{ab, bc, cd, de\}$

• Note that $T(d) = S(d,1)$ and $S(d,|d|) = d$

• Use Jaccard to compute resemblance $r_w(d_1,d_2) = \frac{|S(d_1, w) \cap S(d_2, w)|}{|S(d_1, w) \cup S(d_2, w)|}$