Information Retrieval Overview

DAPI . Information Description, Storage and Retrieval Course
MIEIC, 2020/21 Edition

Sérgio Nunes
DEI, FEUP, U.Porto

Based on Chapters 1, 2, 6, 11, 12 from Introduction to Information Retrieval, Manning, C. et al. (2008)

Background Concepts

Incidence Matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Figure 1.1 A term-document incidence matrix. Matrix element (t, d) is 1 if the play in column d contains the word in row t, and is 0 otherwise.

Boolean Model

- ➔ In the Boolean Retrieval Model queries are represented in the form of a Boolean expression of terms.
- ➔ E.g.: [Brutus AND Caesar AND NOT Calpurnia]
  - ➔ => 110100 AND 110111 AND 101111 = 100100
- ➔ The model views each document as a set of words.
- ➔ Bag of Words (BoW) model, where the exact ordering of terms in a document is ignored and only their presence is considered.

Inverted Index

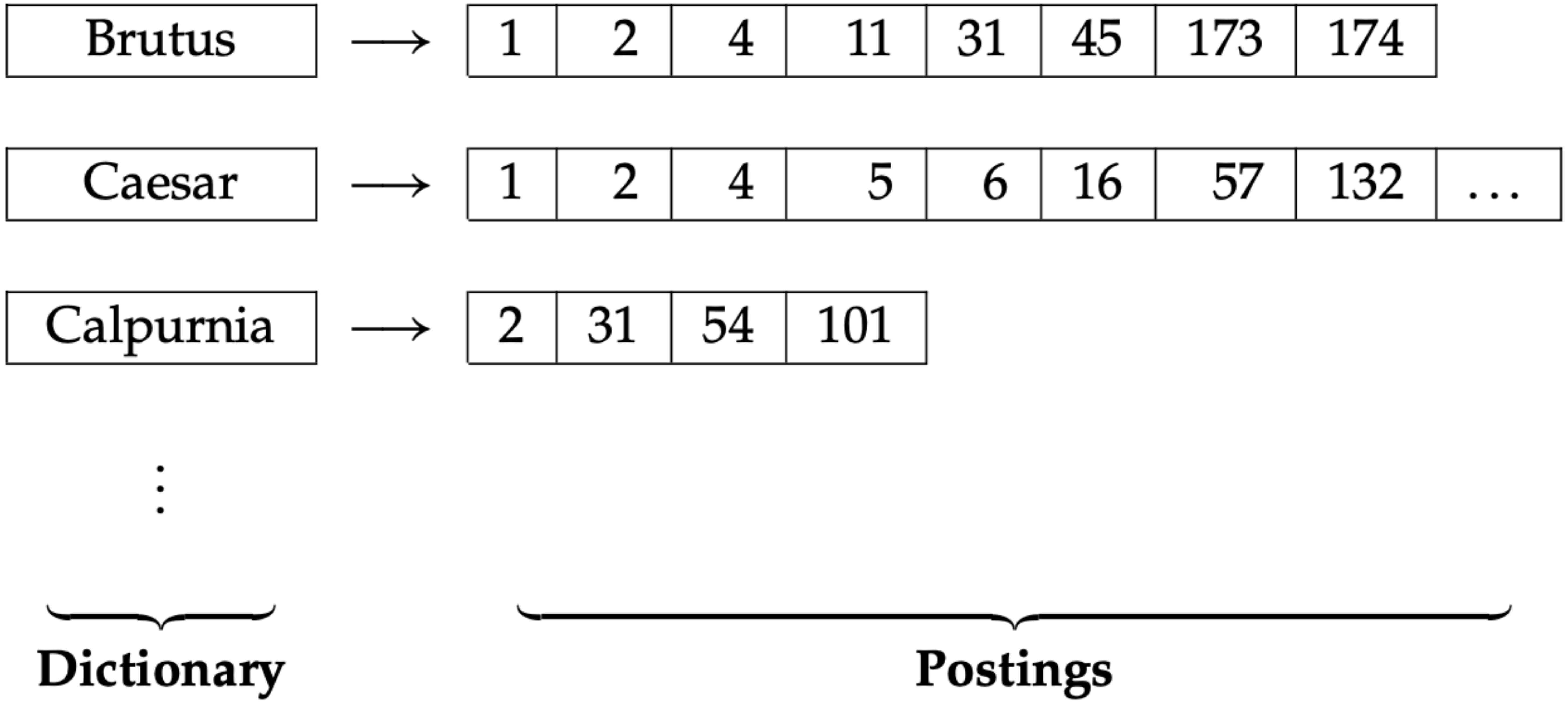

Figure 1.3 The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

Index Construction

- ➔ Choosing the document unit for indexing and the index granularity are important first steps.
- ➔ There is a precision / recall tradeoff in this decision.
  - ➔ "If the units get too small (e.g. sentences), we are likely to miss important passages because terms were distributed over several mini-documents, whereas if units are too large (e.g. books) we tend to get spurious matches and the relevant information is hard for the user to find."

Tokenization

- "Given a character sequence and a defined document unit, tokenization is the task of chopping it into pieces, called tokens."

Input: Friends, Romans, Countrymen, lend me your ears;

Output: $\boxed{\text{Friends}}$ $\boxed{\text{Romans}}$ $\boxed{\text{Countrymen}}$ $\boxed{\text{lend}}$ $\boxed{\text{me}}$ $\boxed{\text{your}}$ $\boxed{\text{ears}}$

- "A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing."

- "A type is the class of all tokens containing the same character sequence."

Stop Words

- ➔ Stop words are words considered of little value in helping select documents in the retrieval process, e.g. a word that exists in all documents.
- ➔ The strategy to determine stop words is by looking the collection frequency of a term (*cf*), i.e. the total number of times a term appears in the document collection.
- ➔ A stop list is a (commonly) hand-picked list of terms to be discarded during the indexing process.

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figure 2.5 A stop list of twenty-five semantically nonselective words that are common in Reuters-RCV1.

Token Normalization

- "Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens."
- Some forms of normalization commonly employed are: accents, capitalization, and stemming and lemmatization for dealing with different forma of a word (e.g. watch, watches, watching).
- Stemming refers to a heuristic process that chops off the ends of words in the hope of reducing inflectional forms.
- Lemmatization refers to the process of reducing inflectional forms by using vocabularies and the morphological analysis of words to find a it's lemma.

Term Weighting

Ranked Retrieval

➔ With the Boolean model a document either matches or does not match a query. In large document collections this is not feasible, thus it is essential for a search system to ranked-order the documents.

➔ Note that there are scenarios where recall is determinant (i.e. all documents need to be analyzed) and thus Boolean search is used.

Parametric and Zone Indexes

- ➔ Although we have considered documents to be a simple sequence of terms, most documents have additional structure (e.g. email message). Additionally, metadata is often associated with a document (e.g. date, authors, title).

- ➔ <u>Parametric indexes</u> are inverted indexes built for specific parameters, or fields, that support parametric search (e.g. "all documents from author Z containing word Y").

- ➔ <u>Zones</u> are a similar concept applied to arbitrary free text (e.g. portion of a document). For example, a document's abstract can be associated to a specific zone index.

Zone Indexes

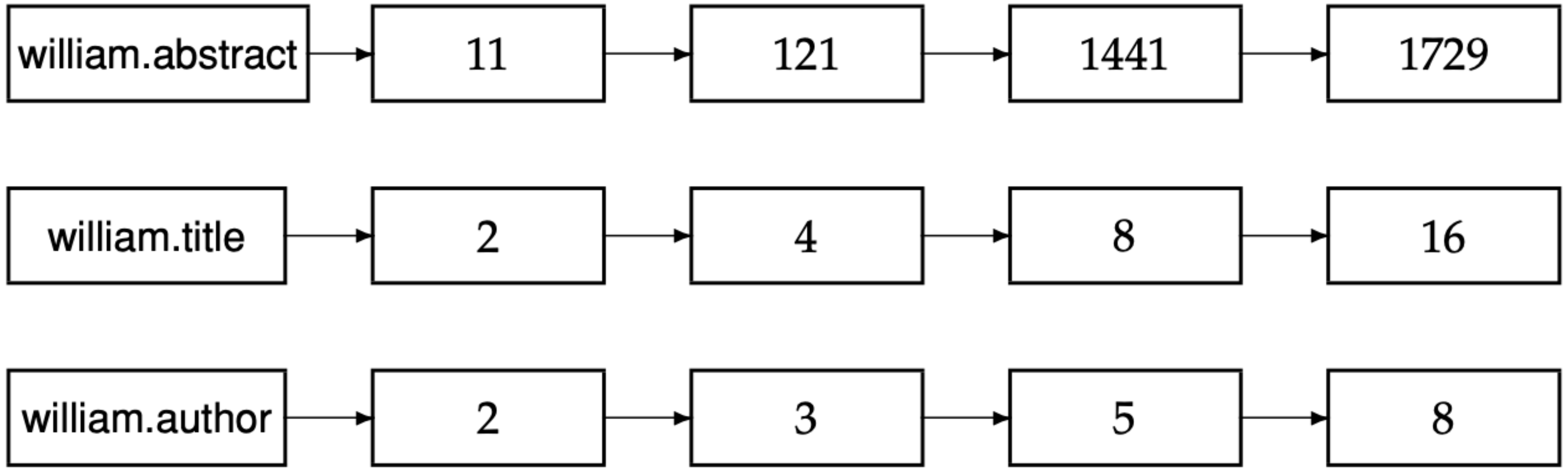

Figure 6.2 Basic zone index; zones are encoded as extensions of dictionary entries.

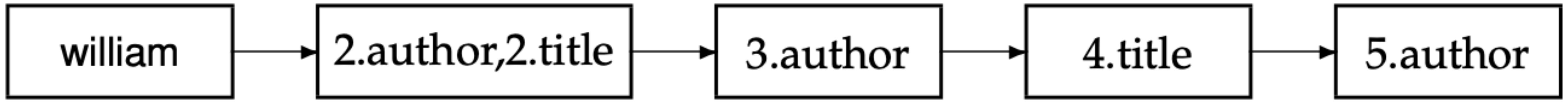

Figure 6.3 Zone index in which the zone is encoded in the postings rather than the dictionary.

Ranked Boolean Retrieval

- ➔ Zones (or fields) can be weighted differently to compute each document's relevance simply using a linear combination of zone scores, where each zone of the document contributes a Boolean value.
- ➔ Zone weights can be specified by an expert (commonly the end user) but are usually learned by the system based on training examples. This method is known as "machine-learned relevance" or "learning to rank".

Term Frequency (tf)

- ➔ The next step is not to consider only the presence or not of a query term in a zone, but the number of mentions of the term (i.e. term frequency).
- ➔ The simplest form of weighting terms differently is simply to assign the weight of a term to the term's frequency.
- ➔ The <u>term frequency</u> of a term in a document is denoted $tf_{t,d}$.
- ➔ In the <u>bag of words model</u>, the ordering is ignored but the number of occurrences of each term is key (in contract with Boolean retrieval).

Document Frequency (df)

- ➔ Raw term frequency suffers from a problem: all terms are considered equally important when assessing a query, when in fact some terms are of little use in determining relevance.
  - ➔ For example, in a collection of thesis dissertations, the term "dissertation" is less likely to be of value since this term probably exists in every document.
- ➔ An important measure to incorporate the <u>discriminative power of a term</u> in a collection is the document frequency of a term (df), i.e. the number of documents that contain a term.

Inverse Document Frequency (idf)

➔ The document frequency of a term is incorporated in the weight of a term by using the concept of inverse document frequency (idf).

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

➔ Where N is the total number of documents in the collection.

➔ The rarer the term is in a collection, the higher it is its idf.

tf-idf

- ➔ Combining term frequency (tf) with inverse document frequency (idf) results in a classical measure in Information Retrieval, the tf-idf weighting scheme.

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

- ➔ tf-idf $_{t,d}$ assigns a term t a weight in a document d that is:
  - ➔ highest when t occurs many times within a small number of documents;
  - ➔ lower when the term occurs fewer times in a document, or occurs in many documents;
  - ➔ lowest when the term occurs in virtually all documents.

Vector Space Model

Vector Space Model

- ➔ The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to number Information Retrieval operations.
- ➔ In a nutshell, each document is represented as a vector, with a component vector for each dictionary term. tf-idf weights are used as components.
- ➔ Thus, the set of documents in a collection may be viewed as a set of vectors in a vector space, in which there is one axis for each term.

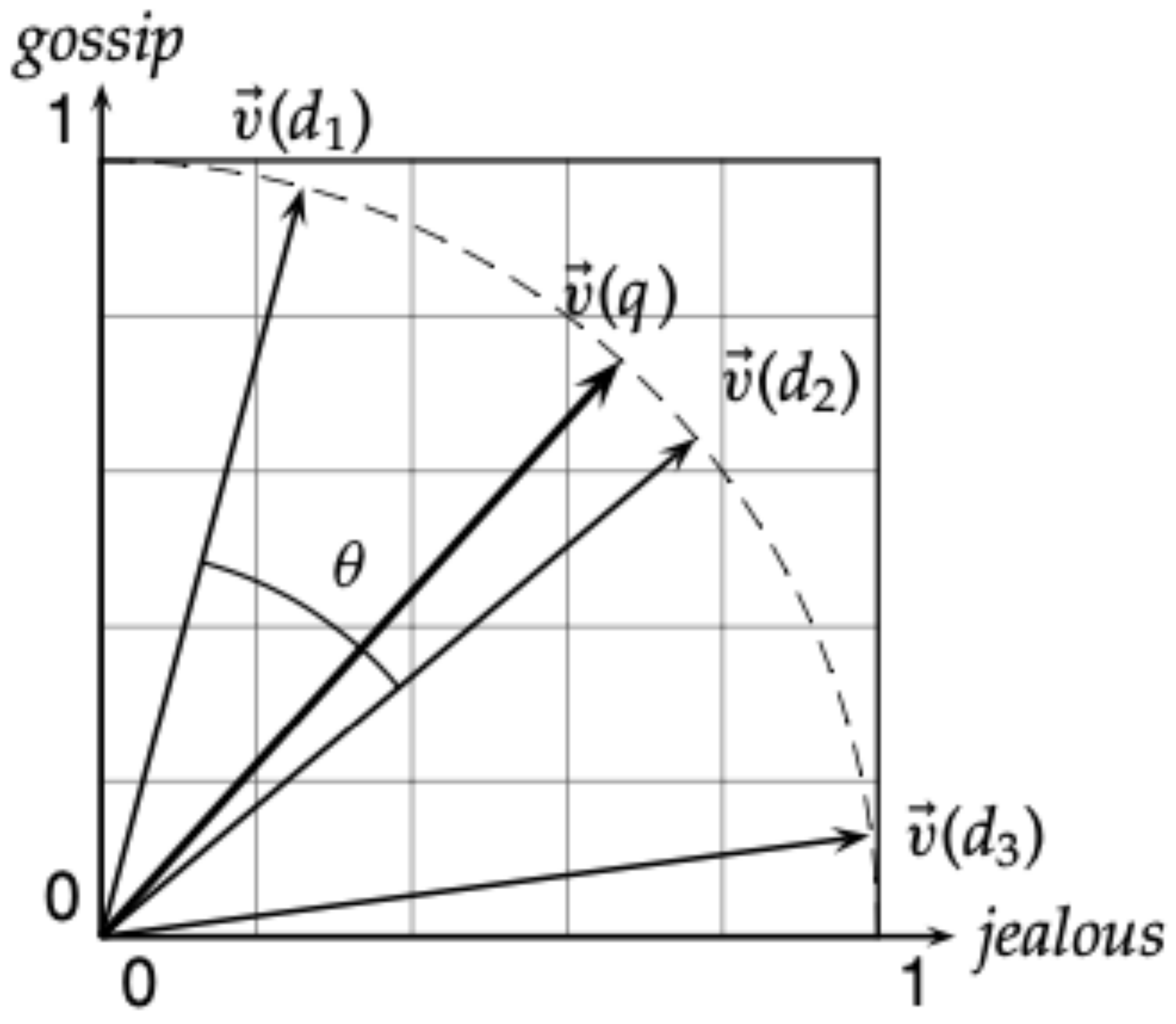

Figure 6.11 Cosine similarity illustrated: $\text{sim}(d_1, d_2) = \cos\theta$.

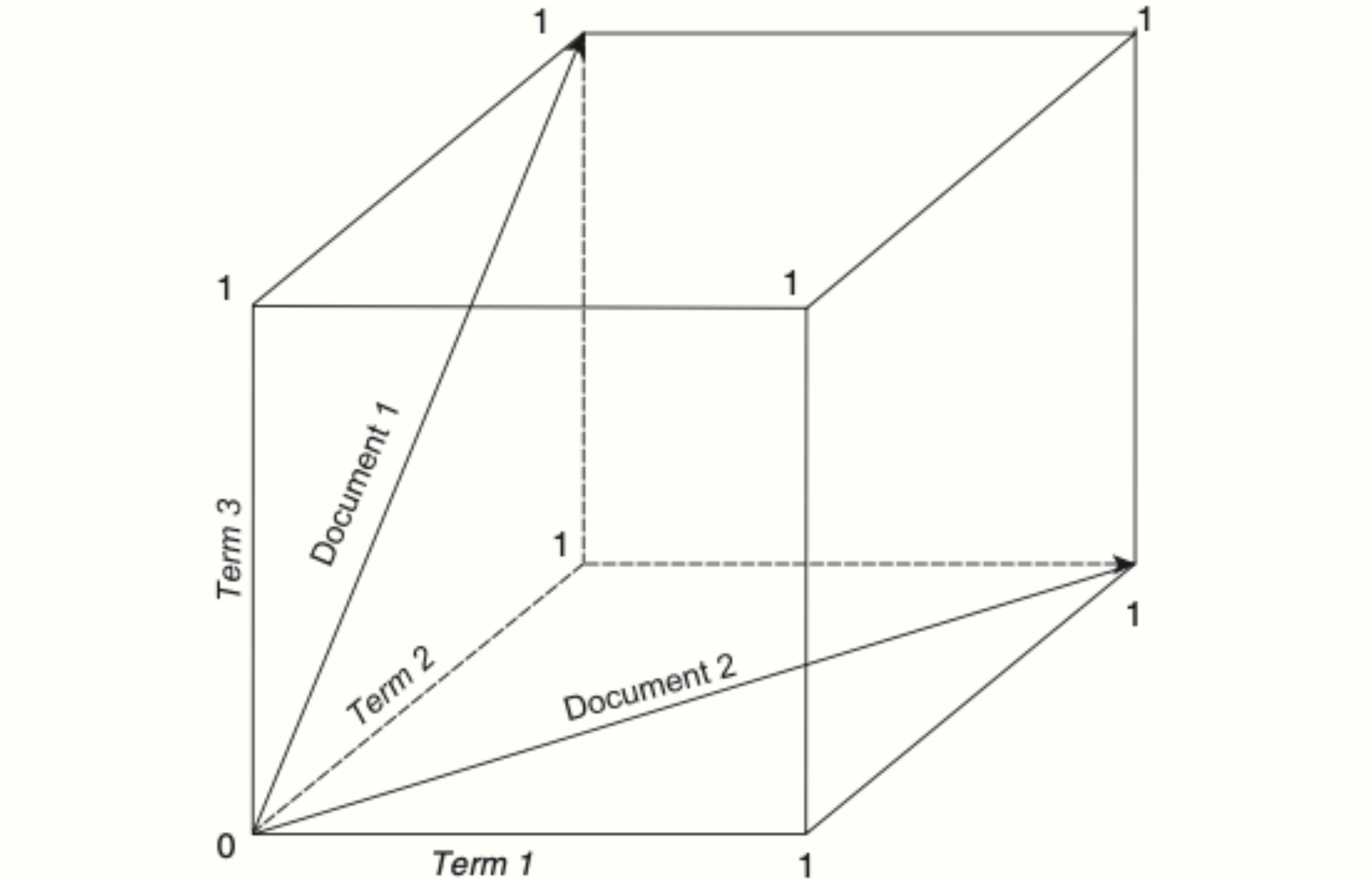

Figure 3.1
Document vectors on the vertices of *the unit hypercube*: each text is represented by a vector whose component, along each term axis, is either 0 or 1. Thus, all vectors must terminate only at one of the corners of the cube. In real life the number of dimensions is far greater than can be illustrated on a two-dimensional page.

Cosine Similarity

- ➔ To quantify the similarity between two documents in this vector space, the cosine similarity of the vector representations of the two documents.
- ➔ In other words, the similarity between two documents is given by the cosine of the angle between the two vector representations of the documents.
- ➔ This approach compensates the effect of document length.

$$\mathrm{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$

Queries as Vectors

- ➔ Queries can also be represented as vectors in a n-dimensional space, being n the number of terms in the query. Basically, queries are viewed as very short documents.
- ➔ The top ranked results for a given query are thus the documents whose vectors have the highest cosine similarity in comparison with the query vector.

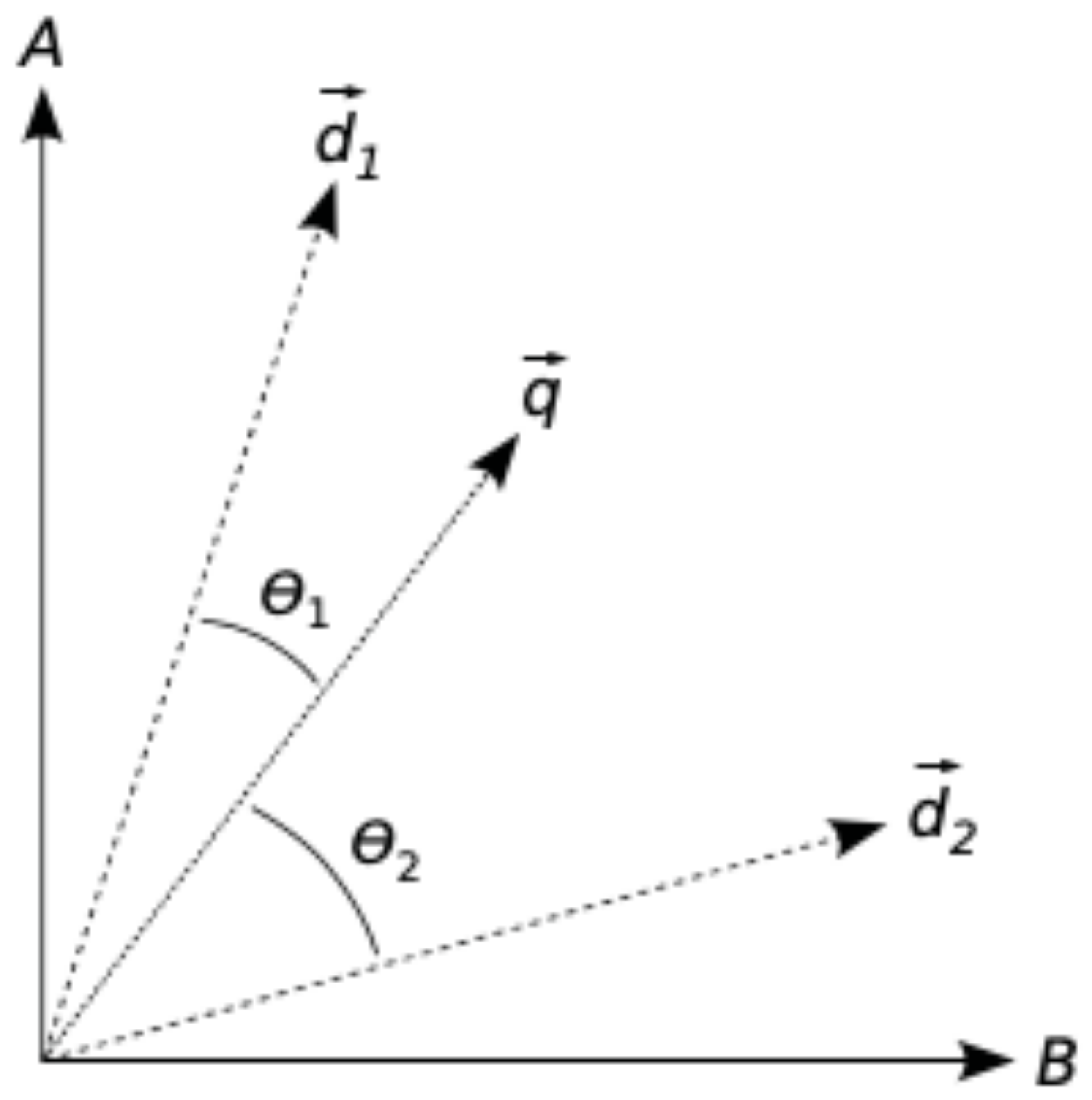

Figure 2.8 Document similarity under the vector space model. Angles are computed between a query vector $\vec{q}$ and two document vectors $\vec{d_1}$ and $\vec{d_2}$. Because $\theta_1 < \theta_2$, d_1 should be ranked higher than d_2.

Language Models for Information Retrieval

Language Models for Information Retrieval

- ➔ Central idea: a document is a good match for a query if the document model is likely to generate the query.
- ➔ In the basic language model approach, a probabilistic language model is built for each document in the collection (M_d).
- ➔ For a given query, documents are ranked based on the probability of the model generating the query: $P(q|M_d)$.

Language Models

➔ A language model is a function that puts a probability measure over strings drawn from some vocabulary.

➔ The sum of all probabilities over a vocabulary for a language model is 1.

➔ The simplest language model, discards all context information (i.e. nearby words), and estimated the probability of each term independently — this is called an <u>unigram model</u>.

  ➔ In this case, the probability of a sequence of terms (e.g. a query) is simply the product of independent term probabilities:

  ➔ $P_{unigram}(t_1 t_2 t_3 t_4) = P(t_1) \times P(t_2) \times P(t_3) \times P(t_4)$

Language Models

- ➔ Bigram language models condition the probability of each term on the previous item:
  - ➔ $P_{bigram}(t_1t_2t_3t_4) = P(t_1) \times P(t_2|t_1) \times P(t_3|t_2) \times P(t_4|t_3)$
- ➔ More complex language models are important in tasks such as speech recognition, spelling correction, or machine translation.
- ➔ In Information Retrieval most language-modeling work uses unigram language models. In IR language models are often estimated from a single model so there is little information to do more.

Example of Language Models

- ➔ D1: Portugal eyes political balance in presidential election
- ➔ D2: After Portuguese elections, Spain braces for elections

- ➔ Unigram Language Models:
  - ➔ M_{d1}: portugal: 0.143 (1/7); eye: 0.143; ...
  - ➔ M_{d2}: after: 0.143; portugal: 0.143; election: 0.286 (2/7); ...

Retrieval based on Language Models

- ➔ Approach for retrieving documents under a language model (LM):
  - ➔ 1. Infer a LM for each document.
  - ➔ 2. Estimate $P(q|M_{di})$, the probability of generating the query according to each of these document models.
  - ➔ 3. Rank the documents according to these probabilities.

Example of Retrieval with Language Models

- ➔ D1: Portugal eyes political balance in presidential election
- ➔ D2: After Portuguese elections, Spain braces for elections

- ➔ Q: [portugal election]
- ➔ $P(q|d_1) = P(portugal|M_{d1}) \times P(election|M_{d1}) = 1/7 \times 1/7 = 0.0204$
- ➔ $P(q|d_2) = P(portugal|M_{d2}) \times P(election|M_{d2}) = 1/7 \times 2/7 = 0.0408$