Tf-idf Weighting

CS5154/6054

Yizong Cheng

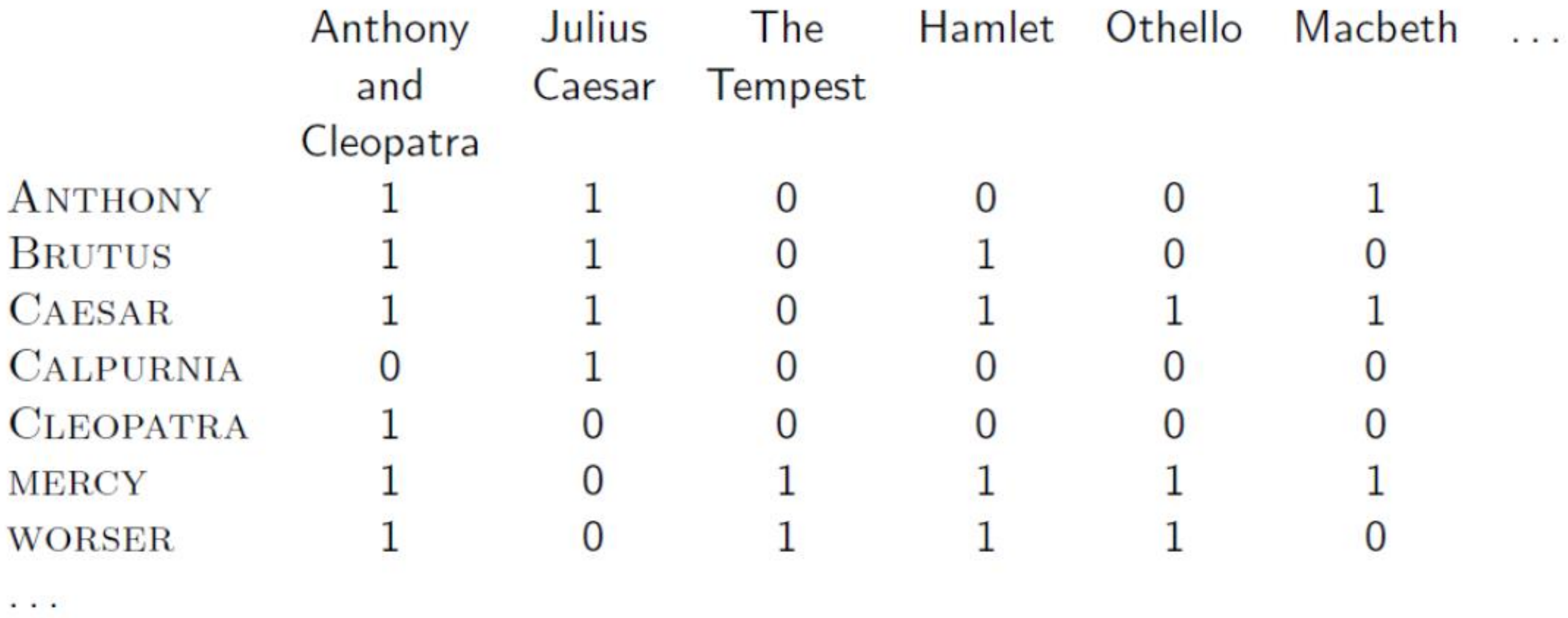
9/6/2022

Today’s Topics

- Collection-dependent document frequency.
- Inverse document frequency (idf) as a measure/weight of the informativeness of a term.
- Context-dependent similarity.

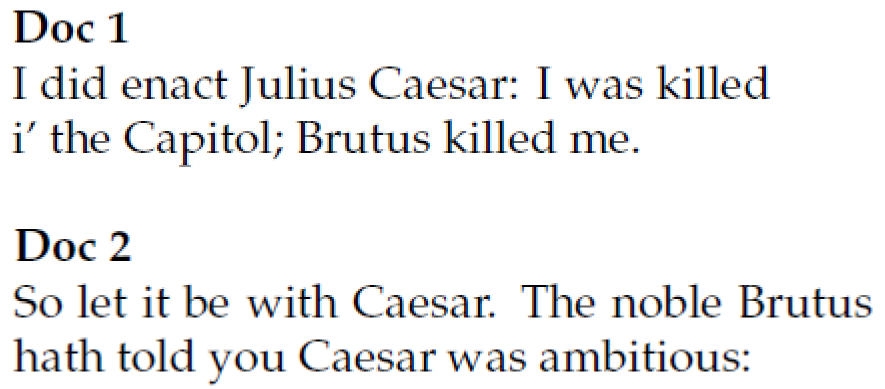
- Term frequency (tf) for each term and each document.
- The count matrix and documents as “bags of words”.
- Tf-idf weight and its variants.

A Document is a Set of Terms



	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							

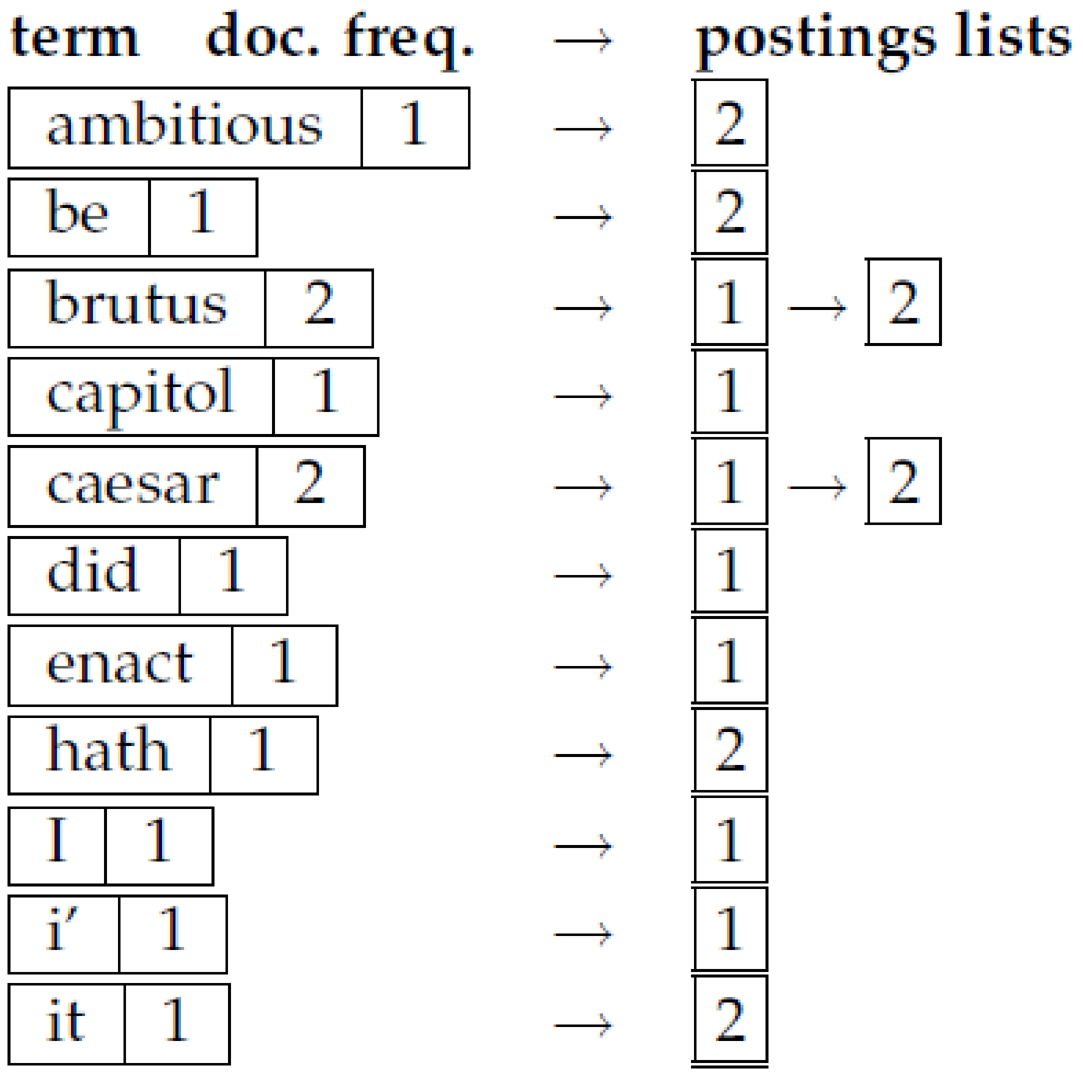
Example Documents



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

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

Document Frequencies of Terms

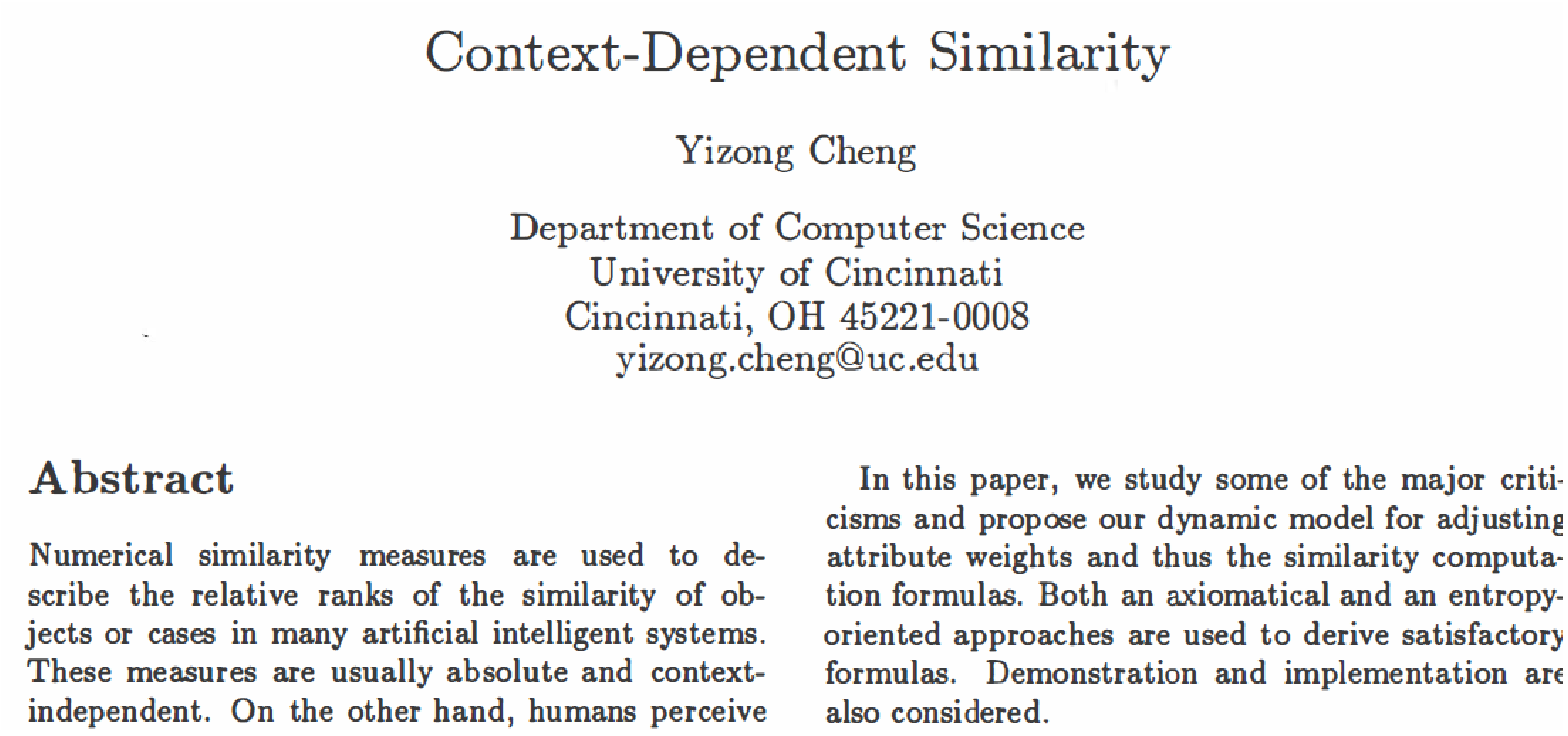


term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i'	1	→	1
it	1	→	2

Document Frequency of a Term

- Document frequency of a term t, df_t , is the number of documents that contains t.
  - It is the size of the set of documents representing the term.
  - It is the length of the postings list for the term in the inverted index.
  - It depends on the collection.

Ranking is Context-dependent (UAI'87)



Context-Dependent Similarity

Yizong Cheng

Department of Computer Science
University of Cincinnati
Cincinnati, OH 45221-0008
yizong.cheng@uc.edu

Abstract

Numerical similarity measures are used to describe the relative ranks of the similarity of objects or cases in many artificial intelligent systems. These measures are usually absolute and context-independent. On the other hand, humans perceive

In this paper, we study some of the major criticisms and propose our dynamic model for adjusting attribute weights and thus the similarity computation formulas. Both an axiomatical and an entropy-oriented approaches are used to derive satisfactory formulas. Demonstration and implementation are also considered.

Tversky and Gati (1978) Experiment: Case 1

- Which of three countries is most similar to Austria?

Sweden

Poland

Hungary

Tversky and Gati (1978) Experiment: Case 1

- Which of three countries is most similar to Austria?

Sweden

Poland

Hungary

Tversky and Gati (1978) Experiment: Case 2

- Which of three countries is most similar to Austria?

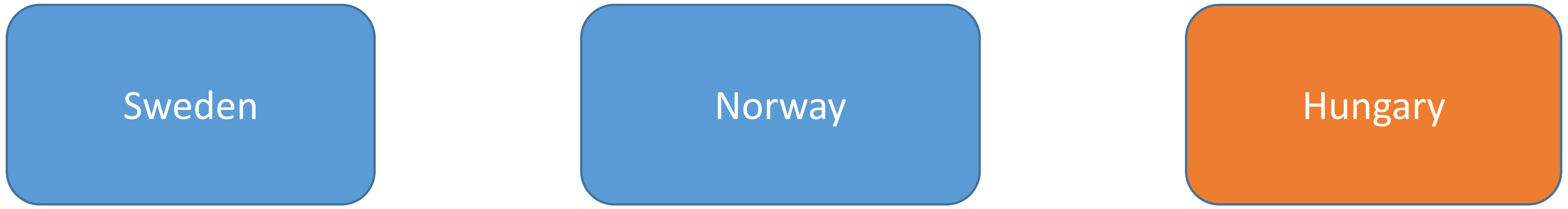
Sweden

Norway

Hungary

Tversky and Gati (1978) Experiment: Case 2

- Which of three countries is most similar to Austria?



Jaccard Coefficient is Context Independent

- Jaccard coefficient is a similarity measure between two sets.
- Documents and queries can be considered as sets of words.
- Jaccard coefficient between a query and a document can be used to rank all documents against a query.
- Jaccard coefficient is document-length sensitive.
  - For the same intersection size |XnY|, the more words in the document, the smaller the Jaccard coefficient is, or the lower the rank is.
- However, Jaccard coefficient and document ranking using has nothing to do with the collection of documents and thus is context independent.

Inverse Document Frequency idf

- df_t is the document frequency, the number of documents that t occurs in.
- df_t is an inverse measure of the informativeness of term t.
- Inverse document frequency, idf_t, is a direct measure of the informativeness of the term.
- The idf weight of term t is defined as follows:

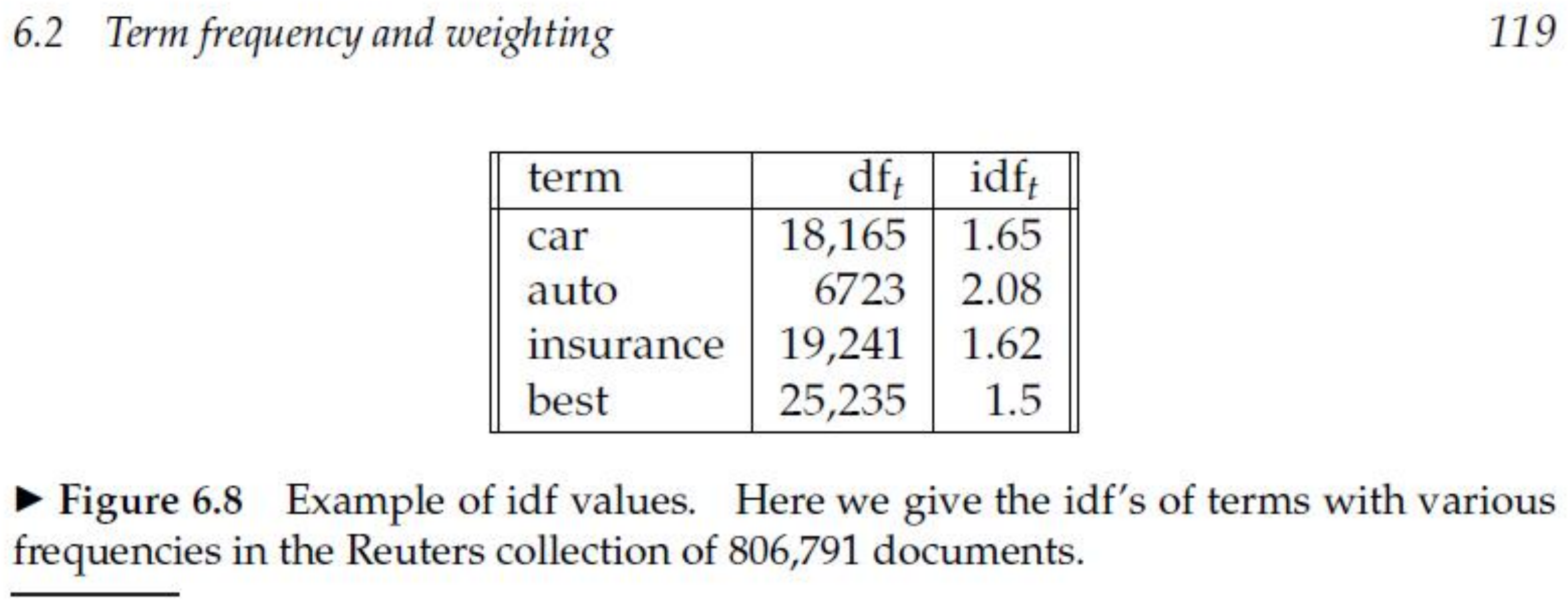
$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

  (N is the number of documents in the collection.)
- $[\log N/\text{df}_t]$ instead of $[N/\text{df}_t]$ to "dampen" the effect of idf

Examples for idf

$$\mathrm{idf}_t = \log_{10} \frac{1{,}000{,}000}{\mathrm{df}_t}$$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0



term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

► **Figure 6.8** Example of idf values. Here we give the idf's of terms with various frequencies in the Reuters collection of 806,791 documents.

Effects of idf on Ranking

- idf gives high weights to rare terms like ARACHNOCENTRIC.
- idf gives low weights to frequent words like GOOD, INCREASE, and LINE.
- idf affects the ranking of documents for queries with 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 little effect on ranking for one-term queries. □

idf is for Context-Dependent Similarities

- Document frequency represents the collection or context.
- idf of a word may be very different in different collections.
- Queries involving words with different idf’s in different collections will result in different rankings of the same document (which may be in different collections).
- tf-idf is the best-known formulation for context-dependent similarity.

Context-Dependent Stop Words

- idf can be used to weight different terms.
- Frequent words may have less weights.
- Stop words become context-dependent.
- No longer need general stop words.

IR3C: Stopwords Defined by df_t

- Certain terms have little or no discriminating power in determining relevance.
  - A collection of documents on the auto industry is likely to have the term **auto** in almost every document.

```
stopwords = set()
for k, v in invertedIndex.items():
    if len(v) > 1000:
        stopwords.add(k)

sets = list(map(lambda s: set(re.findall('\w+', s)) - stopwords, docs))
```

IR5A: Replacing 1 by idf in Intersection Counts

```
N = len(docs)
logN = math.log(N)
query = random.randint(0, N)

intersections = {}
for t in set(re.findall('\w+', docs[query])) :
    idf = logN - math.log(len(invertedIndex.get(t)))
    for d in invertedIndex.get(t):
        if intersections.get(d) == None:
            intersections.update({d : idf})
        else:
            x = intersections.get(d) + idf
            intersections.update({d : x})

res = nlargest(5, intersections, key = intersections.get)
for k in res:
    print(k, intersections.get(k), docs[k])
```

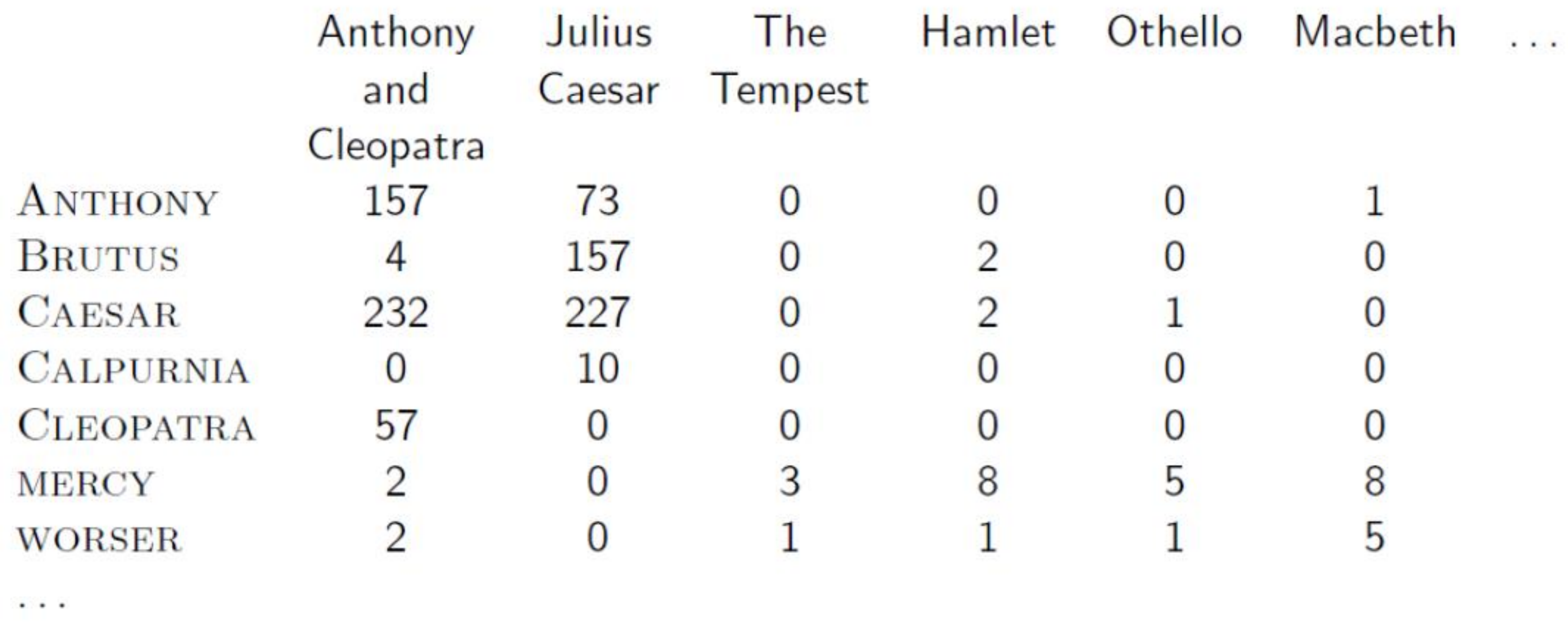
From Intersection Size to Sum of idf Weights

- In counting the intersection sizes between documents and query, a count 1 is replaced by the idf of the term.
- Intersection size (number of terms in the intersection) becomes sum of idf weights of the terms in the intersection.
- Python Counter may no longer work.

Term Frequency and Bag of Words

- We assign to each term in a document a *weight* for that term.
- The simplest approach is to assign the weight to be equal to the number of occurrences of term *t* in document *d*.
- This weighting scheme is referred to as *term frequency* and is denoted $\mathrm{tf}_{t,d}$, with the subscripts denoting the term and the document.
- In this view of a document, known in the literature as the *bag of words model*, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material (in contrast to Boolean retrieval).

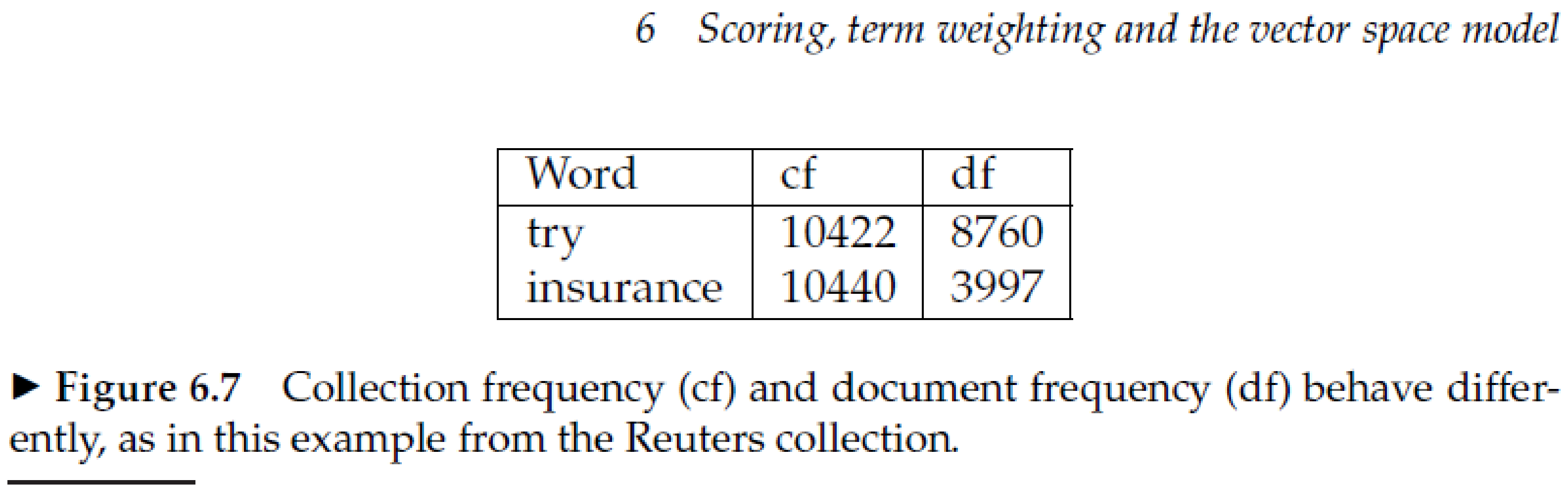
Count Matrix with Term Frequencies



	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	. . .
ANTHONY	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	
. . .							

There is also a Collection Frequency (cf)

- Collection frequency of a term in a collection is the total number of occurrences of a term in the collection.



6 *Scoring, term weighting and the vector space model*

Word	cf	df
try	10422	8760
insurance	10440	3997

▶ **Figure 6.7** Collection frequency (cf) and document frequency (df) behave differently, as in this example from the Reuters collection.

6.2.2 Tf-idf weighting

We now combine the definitions of term frequency and inverse document frequency, to produce a composite weight for each term in each document.

TF-IDF The *tf-idf* weighting scheme assigns to term t a weight in document d given by

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t. \quad (6.8)$$

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

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

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}. \quad (6.9)$$

Tf and Idf for Query

```
query = random.randint(0, N)
print(query, docs[query])
counter.clear()
counter.update(re.findall('\w+', docs[query]))
print(counter)
for t, tf1 in counter.items():
    idf = logN - math.log(len(invertedIndex.get(t)))
    print(t, idf)
```

```
9970 And David longed, and said, Oh that one would give me
drink of the water of the well of Bethlehem, that is at the gate!

Counter({'of': 3, 'the': 3, 'that': 2, 'And': 1, 'David': 1, 'longed': 1,
'and': 1, 'said': 1, 'Oh': 1, 'one': 1, 'would': 1, 'give': 1, 'me': 1,
'drink': 1, 'water': 1, 'well': 1, 'Bethlehem': 1, 'is': 1, 'at': 1,
'gate': 1})
```

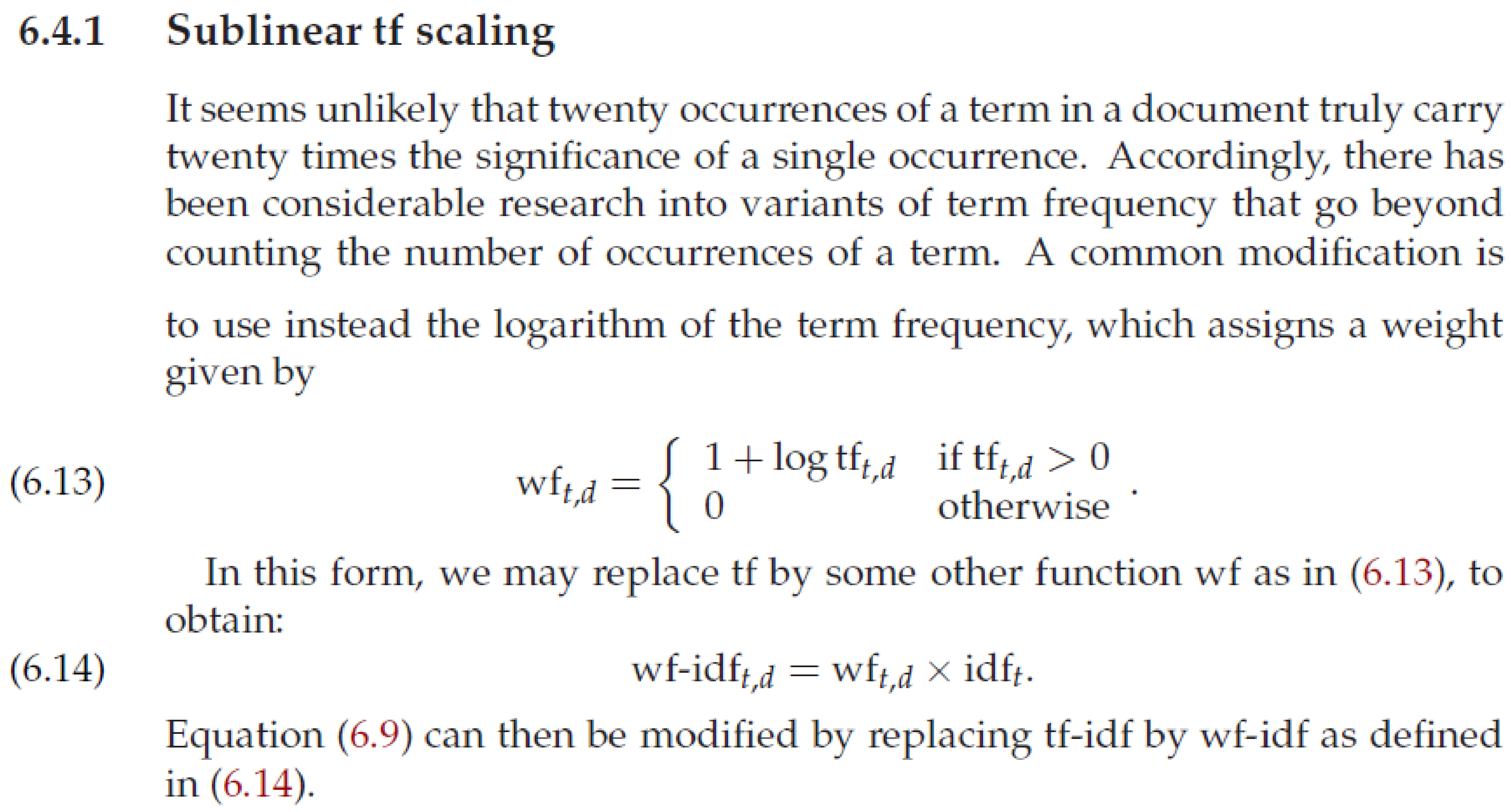
```
And 0.9665054035737519
David 3.5451315280792928
longed 8.24219697877164
and 0.4093894622467751
said 2.187170950173913
Oh 6.766290458962063
that 1.1707297322447197
one 2.967276190029999
would 4.35293096046611
give 3.7299647884428175
me 2.3069721499865334
drink 4.572245534543223
of 0.550768492530187
the 0.2786934196894748
water 4.46085229698561
well 4.8792208099296825
Bethlehem 6.68405236072509
is 1.7491998548092607
at 3.1227072797633024
gate 4.988919727186107
```

IR5B: Tfidf Weights on Both Query and Doc

```
intersections = {}
for t, tf1 in counter.items():
    idf = logN - math.log(len(invertedIndex.get(t)))
    for d, tf2 in invertedIndex.get(t).items():
        tfidf = tf1 * idf * tf2 * idf
        if intersections.get(d) == None:
            intersections.update({d : tfidf})
        else:
            x = intersections.get(d) + tfidf
            intersections.update({d : x})

print()
res = nlargest(5, intersections, key = intersections.get)
for k in res:
    print(k, intersections.get(k), docs[k])
```

Sublinear tf Scaling



6.4.1 Sublinear tf scaling

It seems unlikely that twenty occurrences of a term in a document truly carry twenty times the significance of a single occurrence. Accordingly, there has been considerable research into variants of term frequency that go beyond counting the number of occurrences of a term. A common modification is

to use instead the logarithm of the term frequency, which assigns a weight given by

$$\text{wf}_{t,d} = \begin{cases} 1 + \log \text{tf}_{t,d} & \text{if } \text{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{6.13}$$

In this form, we may replace tf by some other function wf as in (6.13), to obtain:

$$\text{wf-idf}_{t,d} = \text{wf}_{t,d} \times \text{idf}_t. \tag{6.14}$$

Equation (6.9) can then be modified by replacing tf-idf by wf-idf as defined in (6.14).

6.4 Variant tf-idf functions

For assigning a weight for each term in each document, a number of alternatives to tf and tf-idf have been considered. We discuss some of the principal ones here; a more complete development is deferred to Chapter 11. We will summarize these alternatives in Section 6.4.3 (page 128).

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log\frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

▶ **Figure 6.15** SMART notation for tf-idf variants. Here *CharLength* is the number of characters in the document.

SMART Notation

- a mnemonic for representing a specific combination of weights
- The mnemonic for representing a combination of weights takes the form *ddd.qqq* where the first triplet gives the term weighting of the document while the second triplet gives the weighting in the query.
  - The first letter in each triplet specifies the term frequency component of the weighting, the second the document frequency component, and the third the form of normalization used.
- For example, a very standard weighting scheme is *lnc.ltc*, where the document has log-weighted term frequency, no idf (for both effectiveness and efficiency reasons), and cosine normalization, while the query uses log-weighted term frequency, idf weighting, and cosine normalization.

Term Frequency tf

- The term frequency $\mathrm{tf}_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.
- We want to rank documents according to query-document matching scores and use tf as a component in these matching scores.
- But how?
- Raw term frequency is not what we want because:
- A document with $\mathrm{tf} = 10$ occurrences of the term is more relevant than a document with $\mathrm{tf} = 1$ occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

Log Frequency Weighting

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

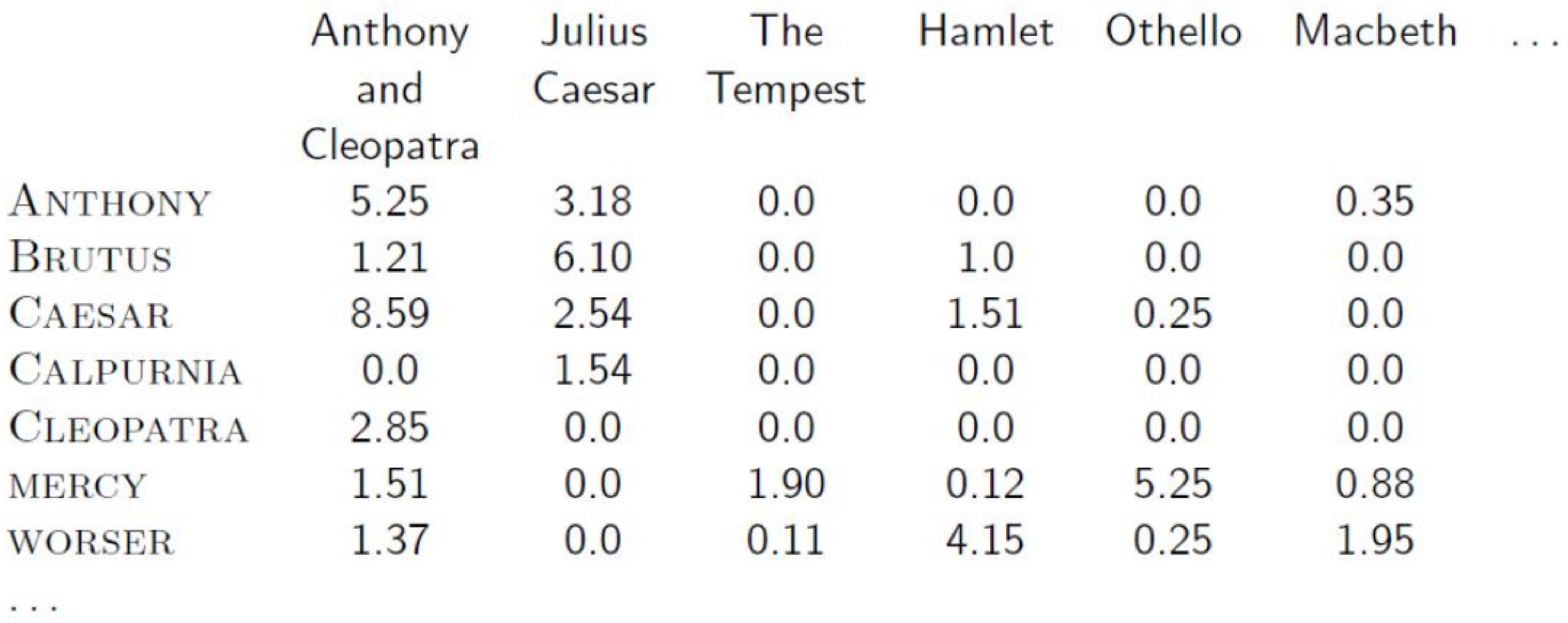
$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$:
  $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Matching score for a document-query pair: sum over terms t in both q and d:
  tf-matching-score$(q, d) = \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$ □

tf-idf Weight

- Assign a tf-idf weight for each term t in each document d: $w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$
- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency component)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency component)

Tf-idf Weight Matrix



	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	. . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35	
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0	
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0	
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0	
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0	
MERCY	1.51	0.0	1.90	0.12	5.25	0.88	
WORSER	1.37	0.0	0.11	4.15	0.25	1.95	
. . .							

Today’s Topics (again)

- Collection-dependent document frequency.
- Inverse document frequency (idf) as a measure/weight of the informativeness of a term.
- Context-dependent similarity.

- Term frequency (tf) for each term and each document.
- The count matrix and documents as “bags of words”.
- Tf-idf weight and its variants.