Intelligent Data Analysis 2020

Lecture 3
Stopping, Stemming & TF-IDF Similarity

Martin Russell

Objectives

- Understand definition and use of **Stop Lists**
- Understand motivation and methods of **Stemming**
- Understand how to calculate the **TF-IDF Similarity** between two documents

Text Pre-Processing

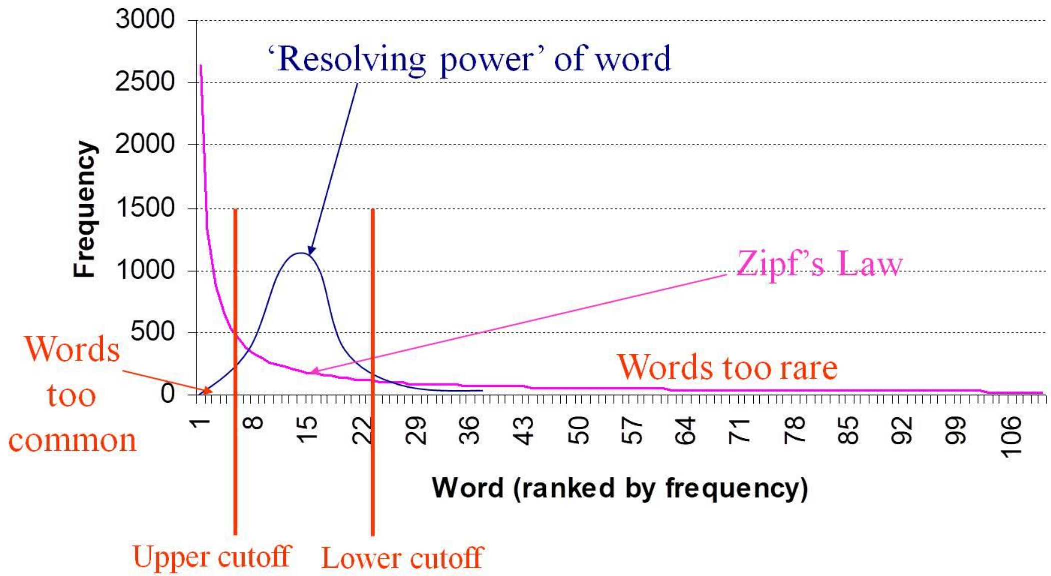
- **Stop Word Removal**: Simple techniques to remove ‘noise words’ from texts
  - Remove common ‘noise’ words which contribute no information to the IR process (e.g. “the”)
- **Stemming**: Remove irrelevant differences from different ‘versions’ of the same word
  - Identify different forms of the same word (e.g. “run” and “ran”) identify them with a common stem
- (Later) Exploit semantic relationships between words
  - If two words have the same meaning, treat them as the same word

Stemming (morphology)

- **Basic idea**: If a query and document contain different forms of the same word, then they are related
- Remove surface markings from words to reveal their basic form:
  - form<u>s</u> → form, form<u>ing</u> → form
  - form<u>ed</u> → form, form<u>er</u> → form
- “form” is the **stem** of forms, forming, formed, former

Stemming (morphology)

- Stemming replaces tokens (words) with **equivalence classes** of tokens (words)
- Equivalence classes are **stems**
  - Reduces the number of different words in a corpus
  - Increases the number of instances of each token



UNIVERSITY OF BIRMINGHAM

Stemming

- Of course, not all words obey simple, regular rules:
  - running $\rightarrow$ run
  - runs $\rightarrow$ run
  - women $\rightarrow$ woman
  - leaves $\rightarrow$ leaf
  - ferries $\rightarrow$ ferry
  - alumnus $\rightarrow$ alumni
  - datum $\rightarrow$ data
  - crisis $\rightarrow$ crises

[Belew, chapter 2]

- Common solution is to identify **sub-pattern of letters** within words and devise **rules** for dealing with these patterns

Stemming

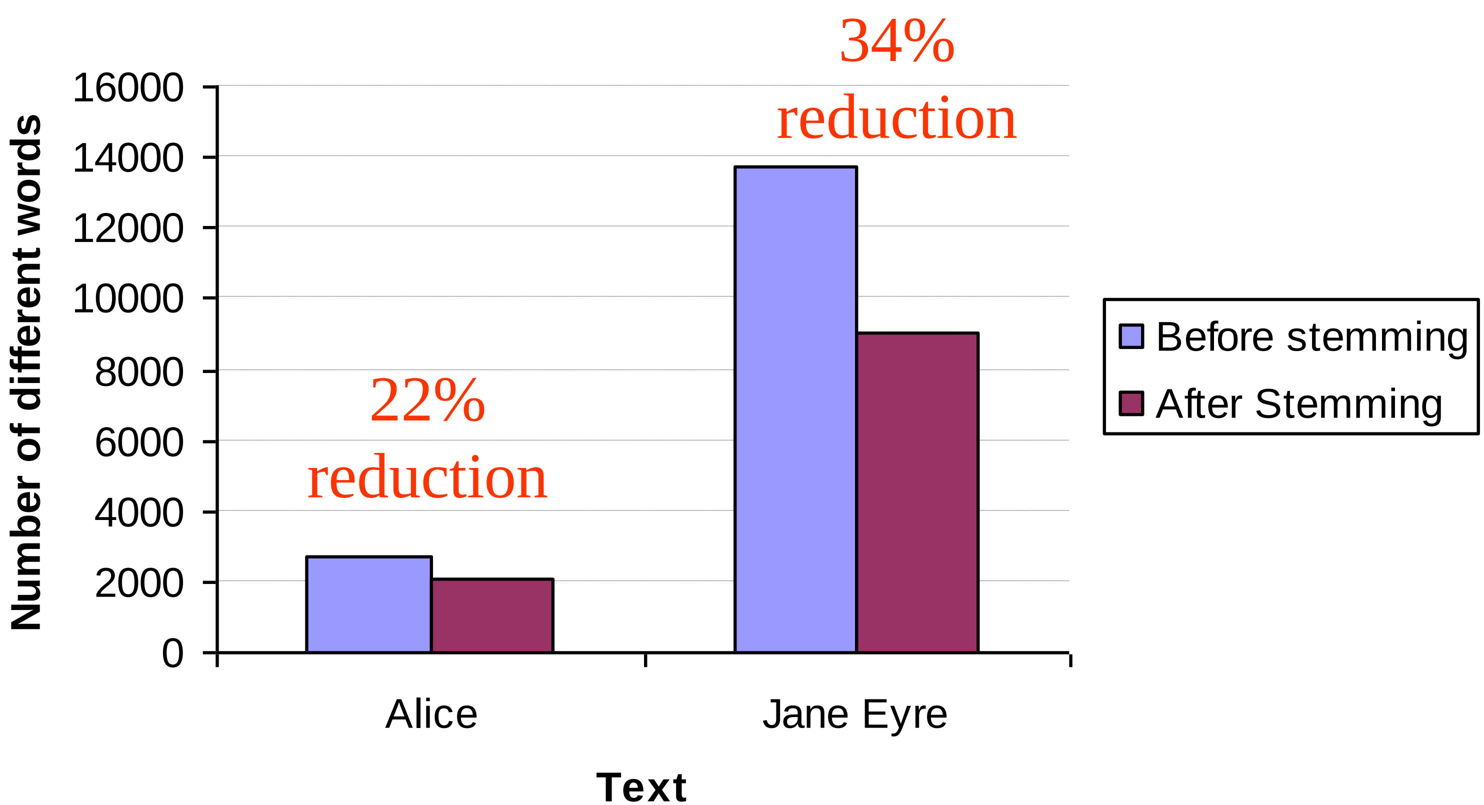
- Example rules [Belew, p 45]
  - (.*)SSES $\rightarrow$ /1SS
    - Any string ending SSES is stemmed by replacing SSES with SS
    - E.G: “classes” $\rightarrow$ “class”
  - (.[AEIOU].*)ED $\rightarrow$ /1
    - Any string containing a vowel and ending in ED is stemmed by removing the ED
    - E.G. “classed” $\rightarrow$ “class”

Stemmers

- A **stemmer** is a piece of software which implements a stemming algorithm
- The **Porter stemmer** is a standard stemmer which is available as a free download
- The Porter stemmer implements a set of about 60 rules
- Use of a stemmer typically reduces vocabulary size by 10% to 50%

Example

- Apply Porter stemmer to *Jane Eyre* and *Alice in Wonderland*



Example

- Examples of results of Porter stemmer:
  - form $\rightarrow$ form
  - former $\rightarrow$ former
  - formed $\rightarrow$ form
  - forming $\rightarrow$ form
  - formal $\rightarrow$ formal
  - formality $\rightarrow$ formal
  - formalism $\rightarrow$ formal
  - formica $\rightarrow$ formica
  - formic $\rightarrow$ formic
  - formant $\rightarrow$ formant
  - format $\rightarrow$ format
  - formation $\rightarrow$ format

Example: First paragraph from 'Alice in Wonderland'

Before

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book', thought Alice 'without pictures or conversation?'

After

alic wa begin to get veri tire of sit by her sister on the bank, and of have noth to do: onc or twice she had peep into the book her sister wa read, but it had no pictur or convers in it, 'and what is the us of a book,' thought alic 'without pictur or convers?'

Noise Words – “Stop words”

There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question

- Noise words in red
  - Vital for the grammatical structure of a text
  - Of little use in the ‘bundle of words’ approach to identifying what a text is “about”

UNIVERSITY OF BIRMINGHAM

Stop Lists

- In Information Retrieval, these words are often referred to as **Stop Words**
- Rather than detecting stop words using rules, stop words are simply specified to the system in a text file: the **Stop List**
- Stop Lists typically consist of the most common words from some large corpus
- There are lots of candidate stop lists online

Example: Short Stop List (50 wds)

the	it	not	her	who
of	with	are	all	will
and	as	but	she	more
to	his	from	there	if
a	on	or	would	out
in	be	have	their	so
that	at	an	we	
is	by	they	him	
was	i	which	been	
he	this	you	has	
for	had	were	when	

The text matters

Alice vs Brown: Most Frequent Words

the	the	as	his	this	an	know	has	thought
and	of	her	on	they	they	them	when	off
to	and	at	be	little	which	like	who	how
a	to	on	at	he	you	were	will	me
she	a	all	by	out	were	again	more	
it	in	with	i	is	her	herself	if	
of	that	had	this	one	all	went	out	
said	is	but	Had	down	she	would	so	
i	was	for	not	up	there	do		
alice	he	so	are	his	would	have		
in	for	be	but	if	their	when		
you	it	not	from	about	we	could		
was	with	very	or	then	him	or		
that	as	what	have	no	been	there		

`stop.c`

- C program on course website
  - Reads in a stop list file (text file, one word per line)
  - Stores stop words in `char **stopList`
  - Read text file one word at a time
  - Compares each word with each stop word
  - Prints out words not in stop list
- `stop stopListFile textFile > opFile`

Examples

Original first paragraph

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,‘ thought Alice `without pictures or conversation?'

Stop list 50 removed

alice beginning get very tired sitting sister bank having nothing do once twice peeped into book sister reading no pictures conversations what use book thought alice without pictures conversation

Stop list Brown removed

alice beginning tired sitting sister bank twice peeped book sister reading pictures conversations book alice pictures

conversation

Simple text retrieval

- Given two documents, d1 and d2
- Let's assume that:
  - All of the stopwords have been removed
  - Stemming has been applied
- We want to know if d1 and d2 are 'about' the same thing – we want to calculate the similarity between d1 and d2
- The simplest approach is to calculate the **TF-IDF similarity**

Matching

- Given a query $\boldsymbol{q}$ and a document $\boldsymbol{d}$ we want to define a number:

$$Sim(\boldsymbol{q},\boldsymbol{d})$$

  which defines the **similarity** between $\boldsymbol{q}$ and $\boldsymbol{d}$
- Given the query $\boldsymbol{q}$ we return documents $\boldsymbol{d_1}\ \boldsymbol{d_2} \ldots \boldsymbol{d_N}$ such that:
  - $\boldsymbol{d_1}$ is the document for which $Sim(\boldsymbol{q},\boldsymbol{d})$ is biggest
  - $\boldsymbol{d_2}$ has the next biggest value of $Sim(\boldsymbol{q},\boldsymbol{d})$,
  - etc

Similarity

- The **similarity** between $\boldsymbol{q}$ and $\boldsymbol{d}$ will depend on the number of **terms** which are common to $\boldsymbol{q}$ and $\boldsymbol{d}$
- But we also need to know how **useful** each common term is for discriminating between different documents.
- For example,
  - It is probably not significant if $\boldsymbol{q}$ and $\boldsymbol{d}$ share "*the*"
  - But it probably is significant if they share "*magnesium*"

IDF weighting

- Popular measure of the significance of a term for discriminating between documents is the **Inverse Document Frequency (IDF)**
- For a token ***t*** define:

if the log $\to$ 0
magnisium log $\to \uparrow$

$$IDF(t) = \log_e\left(\frac{ND}{ND_t}\right)$$

$\ln(ND) - \ln(ND_t)$

- ND is total number of documents in the corpus
- ND_t is number of those documents that include ***t***
- In this context, $\log$ is $\log_e$ – **natural logarithm**

Why is IDF weighting useful?

$$IDF(t) = \log\left(\frac{ND}{ND_t}\right)$$

- Case 1: ***t*** occurs equally often in all documents
  - $ND = ND_{t,}$
  - hence $\boldsymbol{IDF(t) = 0}$
- Case 2: ***t*** occurs in just a few documents
  - $ND > ND_t$
  - hence $\boldsymbol{IDF(t) > 0}$
- Note $IDF(t)$ ignores how often term ***t*** occurs in a document

Effect of Document Length

- Suppose query q consists only of term t
- Suppose document d_1 also consists only of t
  - Number of shared terms is 1
  - Match is ‘perfect’
- Suppose document d_2 has 100 terms, including t
  - Number of shared terms is 1
  - But in this case co-occurrence of t is less significant

UNIVERSITY OF BIRMINGHAM

TF-IDF weight

- Let $\boldsymbol{t}$ be a term and $\boldsymbol{d}$ a document
- TF-IDF – Term Frequency – Inverse Document Frequency
- The TF-IDF **weight** $\boldsymbol{w_{td}}$ of term $\boldsymbol{t}$ for document $\boldsymbol{d}$ is:

$$w_{td} = f_{td} \cdot IDF(t)$$

$freq() \cdot [\ln(ND) - \ln(ND_t)].$

where:

f_{td} = **term frequency** – the number of times $\boldsymbol{t}$ occurs in $\boldsymbol{d}$

TF-IDF weight (continued)

$$w_{td} = f_{td} \cdot IDF(t)$$

- For w_{td} to be large:
  - f_{td} must be large, **so *t* must occur often in *d***
  - $IDF(t)$ must be large, **so *t* must only occur in relatively few documents**

$W_{td} \uparrow : f_{td} \uparrow , IDF(t) \uparrow$

Query weights

- Now suppose t is a term and q is a query. term
- If q is a **long** query, can treat q as a document: ?

$$w_{tq} = f_{tq} \cdot IDF(t)$$

where f_{tq} is the (query) term frequency – the number of times the term t occurs in the query q

- If q is a **short** query, define the TF-IDF weight as

$$w_{tq} = IDF(t)$$

TF-IDF Similarity

- Define the similarity between query ***q*** and document ***d*** as:

Sum over all terms in both ***q*** and ***d***

$t \in q$ and d.

$$Sim(q,d) = \frac{\boxed{\sum_{t \in q \cap d}} w_{td} \cdot w_{tq}}{\boxed{\|d\|} \cdot \boxed{\|q\|}}$$

'Length' of query ***q***

'Length' of document ***d***

Document length

- Suppose $\boldsymbol{d}$ is a document
- For each term $\boldsymbol{t}$ in $\boldsymbol{d}$ we can define the TF-IDF weight $\boldsymbol{w_{td}}$
- The **length** of document $\boldsymbol{d}$ is defined by:

$$Len(d) = \|d\| = \sqrt{\sum_{t \in d} w_{td}^2}$$

weight

UNIVERSITY OF BIRMINGHAM

Comments on Document Length

- This definition of $Len(\boldsymbol{d})$ may not seem very intuitive at first
- It will become more intuitive when we study vector representations of documents and Latent Semantic Indexing (LSI)
- For now, just remember that if $\boldsymbol{x} = (\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{x_3})$ is a vector in 3 dimensional space, then the length of $\boldsymbol{x}$ is given by: $$\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2}$$

Practical Considerations (1)

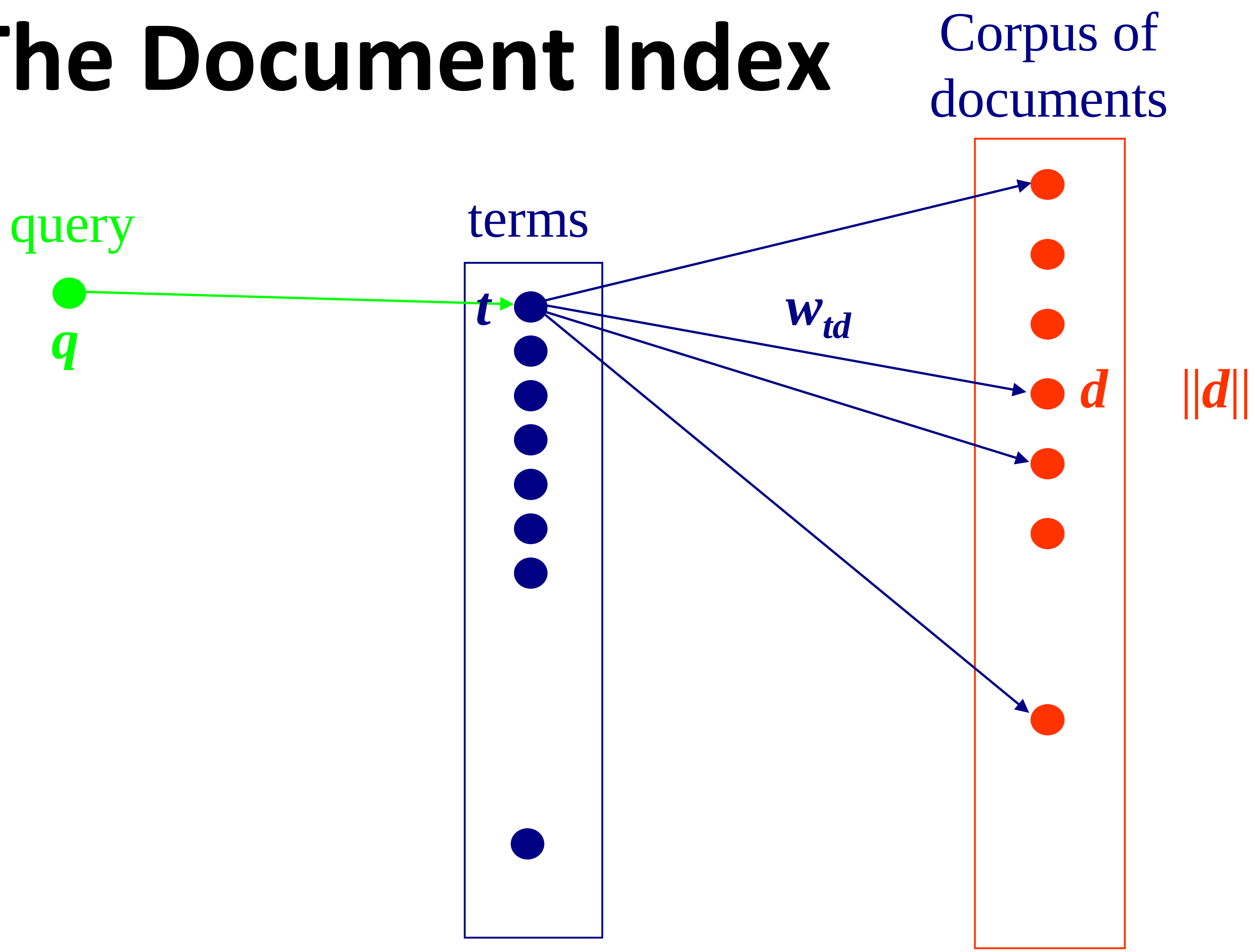
- Given a query $\boldsymbol{q}$:
  - Calculate $\|q\|$ and $\boldsymbol{w_{tq}}$ for each term $\boldsymbol{t}$ in $\boldsymbol{q}$
  - Not too much computation!
- For each document $\boldsymbol{d}$
  - $\|d\|$ can be computed in advance
  - $\boldsymbol{w_{td}}$ can be computed in advance for each term $\boldsymbol{t}$ in $\boldsymbol{d}$
- Potential number of documents is **huge**
- Potential time to compute all $Sim(\boldsymbol{q,d})$ is huge!

Practical Considerations (2)

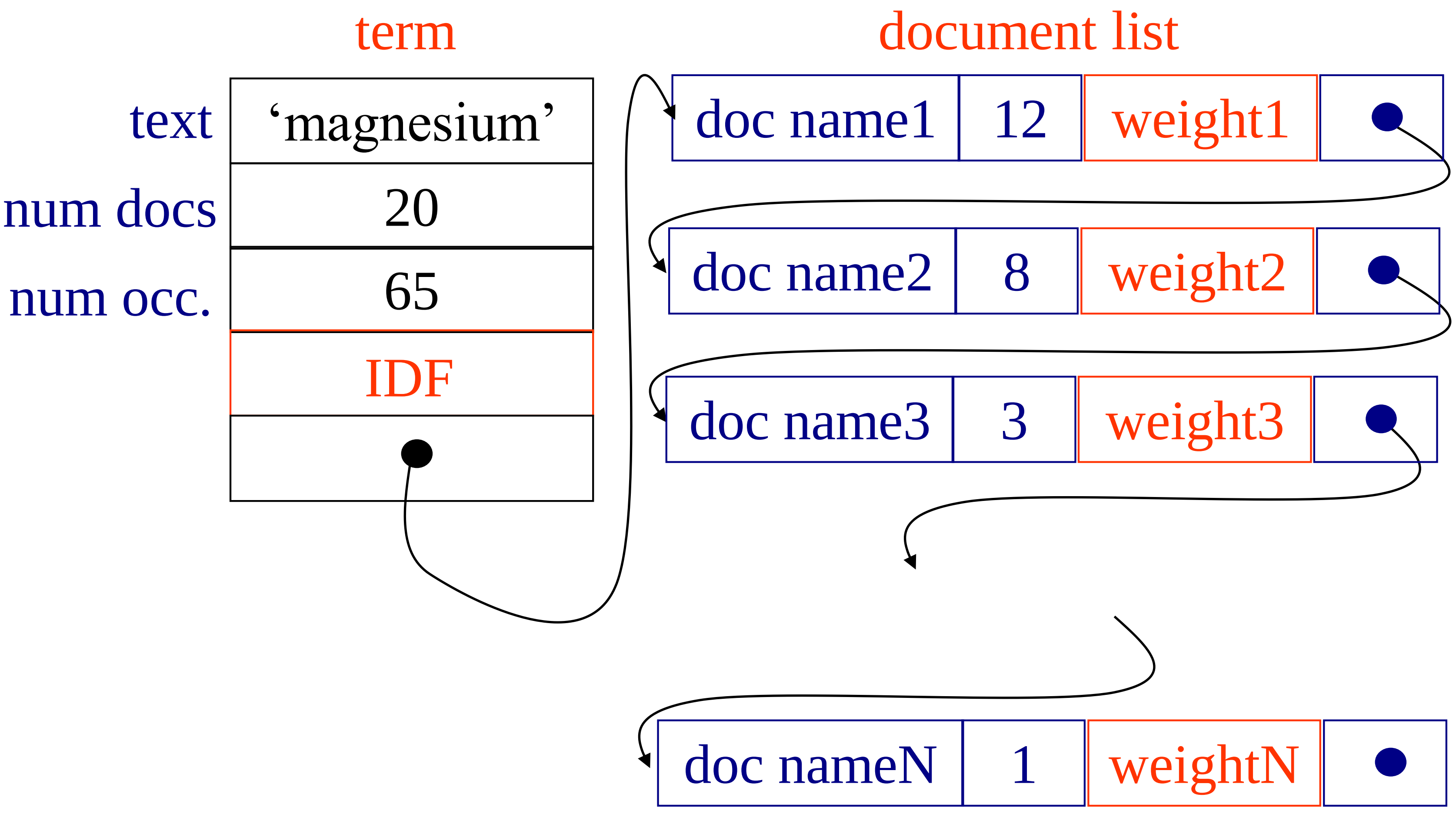
- Suppose the query $\boldsymbol{q}$ contains a term $\boldsymbol{t}$
- If $\boldsymbol{t}$ didn't already occur in the corpus it's of no use
- Need to identify all documents $\boldsymbol{d}$ which include $\boldsymbol{t}$

  (so that we can calculate $Sim(\boldsymbol{q},\boldsymbol{d})$ for these $\boldsymbol{d}$)
- This will take too long if the number of documents is very large (as it will be in real applications)
- To speed up this computation, we compute a data structure, called the Document Index, in advance

The Document Index



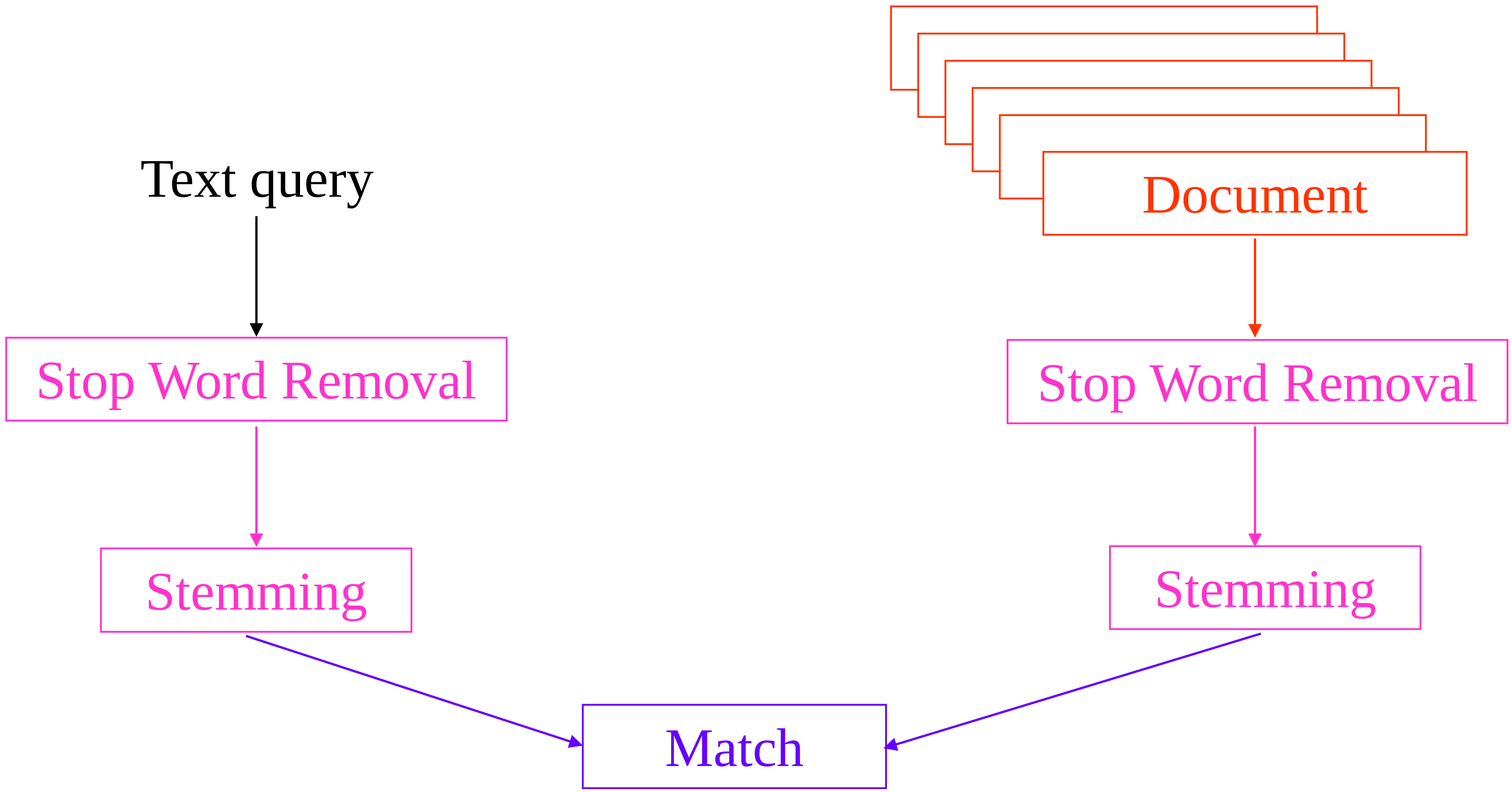
The Document Index



Practical considerations

- Order **terms** according to decreasing IDF
- For each term, order **documents** according to decreasing weight
- For each term in the query
  - Identify term in index
  - Increment similarity scores for documents in the list for this term
  - Option to stop when weight falls below some threshold

Summary of the IR process



Homework

- Download Porter Stemmer from the web
  - See URL on course web page
  - Compile and run it under your favourite OS
  - Try it out on some words and text corpora
- Download `stop.c` from the web
  - Download some stop lists
  - Compile and run `stop.c` under your favourite OS
  - Try it out on some stop lists and text corpora
- How can you make `stop.c` run on stemmed text?

Summary

- Stemming and stop-word removal
- TF-IDF similarity
- Practical considerations – the document index