

Structural Text Features

CISC489/689-010, Lecture #13

Monday, April 6th

Ben Carterette


Structural Features

- So far we have mainly focused on “vanilla” features of terms in documents
 - Term frequency, document frequency
 - “Bag of words” models
- Some documents have *structure* that we could leverage for improved retrieval
 - Natural language has structure as well
- We can derive features from this structure, especially from the placement of terms within structure or placement of terms with respect to each other

Example: HTML

- “HyperText Markup Language”
- Provides document structure using tags enclosing text
 - `<title>`: enclosed text displayed at top of browser
 - `<body>`: enclosed text displayed in browser
 - `<h1>`: enclosed text displayed in large font
 - ``: enclosed text displayed in bold
 - `<a>`: enclosed text can be clicked to go to another page
- The text enclosed in fields is often unstructured or structured with more HTML

Example: HTML



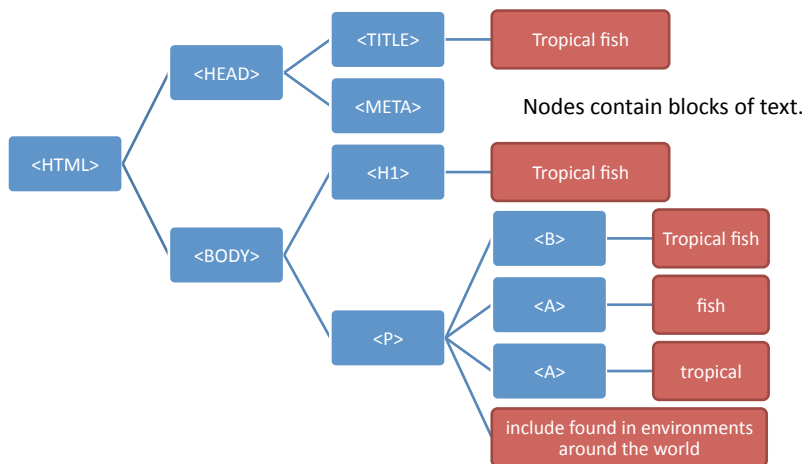
```

<html>
<head>
<meta name="keywords" content="Tropical fish, Airstone, Albinism, Algae eater,
Aquarium, Aquarium fish feeder, Aquarium furniture, Aquascaping, Bath treatment
(fishkeeping), Berlin Method, Biotope" />
...
<title>Tropical fish - Wikipedia, the free encyclopedia</title>
</head>
<body>
...
<h1 class="firstHeading">Tropical fish</h1>
...
<b>Tropical fish</b> are found in <a href="/wiki/Fish" title="Fish">fish</a> found in <a
href="/wiki/Tropics" title="Tropics">tropical</a> environments around the world,
including both <a href="/wiki/Fresh_water" title="Fresh water">freshwater</a> and <a
href="/wiki/Sea_water" title="Sea water">salt water</a> species. <a
href="/wiki/Fishkeeping" title="Fishkeeping">Fishkeepers</a> often use the term
<i>tropical fish</i> to refer only those requiring fresh water, with saltwater tropical fish
referred to as <i><a href="/wiki/List_of_marine_aquarium_fish_species" title="List of
marine aquarium fish species">marine fish</a></i></p>
<p>Tropical fish are popular <a href="/wiki/Aquarium" title="Aquarium">aquarium</a>
fish , due to their often bright coloration. In freshwater fish, this coloration typically
derives from <a href="/wiki/Iridescence" title="Iridescence">iridescence</a>, while salt
water fish are generally <a href="/wiki/Pigment" title="Pigment">pigmented</a>.</p>
...
</body></html>

```

Example: HTML

- HTML pages organize into trees.



Example: Email

- Header fields provide some structure

From: Ed <edward@yahoo.com>
Subject: [SIG-IRList] ICAI-09 Call for papers: special session on Semantic Web and Information Retrieval
Date: April 1, 2009 4:38:50 AM EDT
To: IRList@lists.shf.ac.uk

ICAI-09 Call for papers: special session on Semantic Web and Information Retrieval

The **4th Indian International Conference on Artificial Intelligence (ICAI-09)** will be held in Tumkur (near Bangalore), India during December 16-18 2009. The conference consists of paper presentations, special workshops, sessions, invited talks and local tours, etc. We invite draft paper submissions. Please see the website: <http://www.iiconference.org> for more details of the conference.

Sincerely

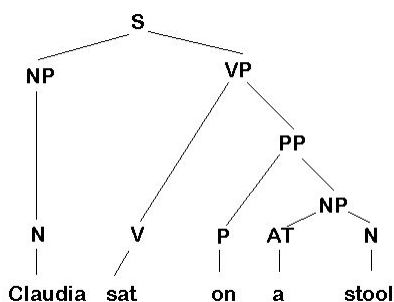
Ed
Publicity Committee

This SIGIR-IRList message and the SIG-IRList Digest (a moderated IR newsletter), are brought

[illegible]

Structure in Natural Language

- One example: parse trees



(from <http://www.lancs.ac.uk/fss/courses/ling/corpus/Corpus2/2PARSE.HTM>)

Hyper-Structure

- The documents themselves may occur within some structure
 - The web: documents link to each other, creating a graph structure
 - Email: threaded conversations
 - Sentences form paragraphs, paragraphs form sections, sections form chapters, chapters form books, ...
- This structure may provide useful features

Using Structural Features in Retrieval

- Steps:
 - Derive features – document processing
 - Index features – using inverted lists
 - Retrieval using features – retrieval models, scoring functions, query languages

Specific Features

- Phrases:
 - Sequences of words in order
 - Users want to query phrases, e.g. “tropical fish”
- Fields and tags:
 - Markup enclosing parts of documents
 - We want to emphasize some parts, de-emphasize others. E.g. titles important, sidebars not
- Web hyper-structure:
 - Links between pages
 - We want pages that are frequently linked using the same text to score higher for queries that contain that text
- What are the features, how do we derive them, how do we store them, and how do we model them in retrieval?

Deriving and Indexing Features

- Derivation considerations:
 - Computational time and space requirements
 - Errors in processing
 - Use in queries
- Indexing considerations:
 - Fast query processing
 - Flexibility (index once with all info for calculating anything you can imagine vs. re-index every time you come up with a new idea)
 - Storage

Phrases

- Many queries are 2-3 word phrases
- Phrases are
 - More precise than single words
 - e.g., documents containing “black sea” vs. two words “black” and “sea”
 - Less ambiguous
 - e.g., “big apple” vs. “apple”
- Can be difficult for ranking
 - e.g., Given query “fishing supplies”, how do we score documents with
 - exact phrase many times, exact phrase just once, individual words in same sentence, same paragraph, whole document, variations on words?

Phrases

- Text processing issue – how are phrases recognized?
- Three possible approaches:
 - Identify syntactic phrases using a *part-of-speech* (POS) tagger
 - Use word *n-grams*
 - Store word positions in indexes and use *proximity operators* in queries

POS Tagging

- POS taggers use statistical models of text to predict syntactic tags of words
 - Example tags:
 - NN (singular noun), NNS (plural noun), VB (verb), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).
- Phrases can then be defined as simple noun groups, for example

Pos Tagging Example

Original text:

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger:

Document/NN will/MD describe/VB marketing/NN strategies/NNS carried/VBD out/IN by/IN U.S./NNP companies/NNS for/IN their/PRP agricultural/JJ chemicals/NNS ./, report/NN predictions/NNS for/IN market/NN share/NN of/IN such/JJ chemicals/NNS ./, or/CC report/NN market/NN statistics/NNS for/IN agrochemicals/NNS ./, pesticide/NN ./, herbicide/NN ./, fungicide/NN ./, insecticide/NN ./, fertilizer/NN ./, predicted/VBN sales/NNS ./, market/NN share/NN ./, stimulate/VB demand/NN ./, price/NN cut/NN ./, volume/NN of/IN sales/NNS ./.

Example Noun Phrases

TREC data		Patent data	
Frequency	Phrase	Frequency	Phrase
65824	united states	975362	present invention
61327	article type	191625	u.s. pat
33864	los angeles	147352	preferred embodiment
18062	hong kong	95097	carbon atoms
17788	north korea	87903	group consisting
17308	new york	81809	room temperature
15513	san diego	78458	seq id
15009	orange county	75850	brief description
12869	prime minister	66407	prior art
12799	first time	59828	perspective view
12067	soviet union	58724	first embodiment
10811	russian federation	56715	reaction mixture
9912	united nations	54619	detailed description
8127	southern california	54117	ethyl acetate
7640	south korea	52195	example 1
7620	end recording	52003	block diagram
7524	european union	46299	second embodiment
7436	south africa	41694	accompanying drawings
7362	san francisco	40554	output signal
7086	news conference	37911	first end
6792	city council	35827	second end
6348	middle east	34881	appended claims
6157	peace process	33947	distal end
5955	human rights	32338	cross-sectional view
5837	white house	30193	outer surface

Noun Phrase Inverted Lists

united → (155833, (d_1 , 17), (d_3 , 22), ...)
 states → (209821, (d_1 , 48), (d_2 , 5), (d_3 , 24), ...)
 united states → (65824, (d_1 , 15), (d_3 , 20), ...)

Q = "united states": retrieve inverted list for phrase "united states" and process

Q = united states: retrieve inverted lists for terms "united", "states" and process

Word N-Grams

- POS tagging too slow for large collections
- Simpler definition – phrase is any sequence of n words – known as *n-grams*
 - *bigram*: 2 word sequence, *trigram*: 3 word sequence, *unigram*: single words
 - N-grams also used at character level for applications such as OCR
- N-grams typically formed from *overlapping* sequences of words
 - i.e. move n -word "window" one word at a time in document

Word Bigrams



Tropical fish
fish include
include fish
fish found
found in
in tropical
tropical environments
environments around
around the
the world
...

Bigram Inverted Lists

tropical fish $\rightarrow (23121, (d_1, 8), (d_2, 6), \dots)$

fish include $\rightarrow (4, (d_1, 1), (d_9, 1), \dots)$

the world $\rightarrow (59434, (d_1, 1), (d_2, 2), \dots)$

Though many unusual phrases are included, term statistics help ensure that they do not hurt retrieval

N-Grams

- Frequent n-grams are more likely to be meaningful phrases
- N-grams form a Zipf distribution
 - Better fit than words alone
- Could index all n-grams up to specified length
 - Much faster than POS tagging
 - Uses a lot of storage
 - e.g., document containing 1,000 words would contain 3,990 instances of word n-grams of length $2 \leq n \leq 5$

Google N-Grams

- Web search engines index n-grams
- Google sample:

Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663
- Most frequent trigram in English is “all rights reserved”
 - In Chinese, “limited liability corporation”

Use Term Positions

- Rather than store phrases in index directly, store term positions and locate phrases at query time
- Match phrases or words within a window
 - e.g., "tropical fish", or "find tropical within 5 words of fish"

tropical	1,1		1,7	2,6	2,17		3,1		
fish	1,2	1,4		2,7	2,18	2,23	3,2	3,6	4,3 4,13

Phrase Method Tradeoffs

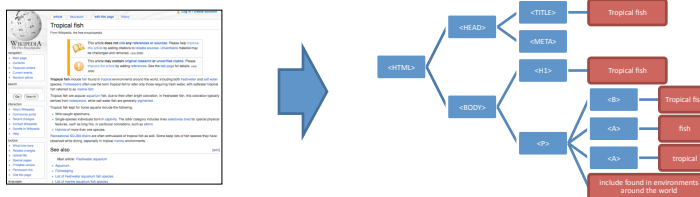
- POS tagging:
 - Very long index time, possible errors, medium storage requirement, not very flexible
 - Fast phrase-query processing
- N-Grams:
 - High storage requirement
 - More flexible, fast phrase-query processing
- Term positions:
 - Medium-low storage requirement, very flexible
 - Possibly slower query processing due to needing to calculate collection statistics

Parsing

- Basic parsing: identify which parts of documents to index, which to ignore
- Full parsing: identify and label parts of documents, maintain structure, decide which parts are relatively more important

HTML Parsing

- An HTML parser produces a DOM tree



- We want to store basic term information (tf, idf) as well as information about the nodes the term appears in

Indexing Fields

- After parsing we have:
 - <title>: tropical fish
 - <body>: tropical fish tropical fish include fish found in tropical environments around the world ...
 - <h1>: tropical fish
 - : tropical fish
 - <a>: fish
 - <a>: topical
- Ideas for indexing:
 - Store field information in inverted list.
 - Add new inverted lists for fields.
 - Use *extents* to keep track of fields in documents.

Field Information in Inverted Lists

- Creating the term inverted list:
 - For each document the term appears in,
 - For each field the term appears in in that document,
 - Store the term frequency within the field
- Also store the “field frequency”
 - i.e. total number of times the term appears in each field through the collection

Field Information in Inverted List

$$t_i \rightarrow (df_i, ff_{i1}, ff_{i2}, \dots, (d_1, tf_{i1}, (f_1, tff_{i,1,1}), (f_2, tff_{i,1,2})), \dots, (d_j, tf_{ij}, (f_k, tff_{ijk})))$$

df_i = Document frequency of term i

ff_{ik} = Field frequency of term i in field k

d_j = Document number j

tf_{ij} = Term frequency of term i in document j

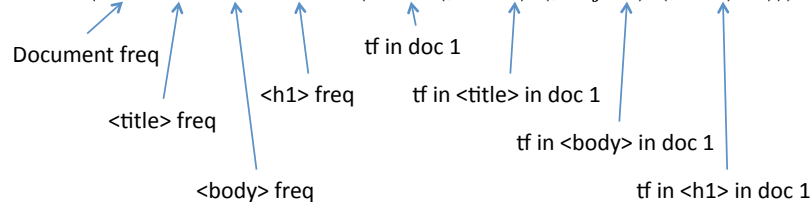
f_k = Field number k

tff_{ijk} = Term frequency of term i in field k in document j

Example

tropical $\rightarrow (29144, 25, 29142, 299, \dots, (d_1, 20, (f_{title}, 1), (f_{body}, 19), f_{h1}, 3), \dots))$

fish $\rightarrow (41002, 89, 40955, 865, \dots, (d_1, 35, (f_{title}, 1), (f_{body}, 34), (f_{h1}, 5), \dots))$



Add New Inverted Lists

- Instead of storing all field information in one list, create a new list for each field the term appears in

$$t_i \rightarrow (df_i, (d_1, tf_{i1}), \dots, (d_j, tf_{ij}))$$

$$t_i.f_k \rightarrow (ff_{ik}, (d_1, tff_{i1k}), \dots, (d_j, tff_{ijk}))$$

- Adds K new inverted lists, where K = the total number of fields the term appears in.

Example

$$\begin{aligned} \text{tropical} &\rightarrow (29144, (d_1, 20), \dots) \\ \text{tropical.title} &\rightarrow (25, (d_1, 1), \dots) \\ \text{tropical.body} &\rightarrow (29142, (d_1, 19), \dots) \\ \text{tropical.h1} &\rightarrow (299, (d_1, 3), \dots) \end{aligned}$$

$$\begin{aligned} \text{fish} &\rightarrow (41002, (d_1, 35), \dots) \\ \text{fish.title} &\rightarrow (89, (d_1, 1), \dots) \\ \text{fish.body} &\rightarrow (40955, (d_1, 34), \dots) \\ \text{fish.h1} &\rightarrow (865, (d_1, 5), \dots) \end{aligned}$$

Extents

- An *extent* is a contiguous region in a document
- Defined by a starting term position and an ending term position



Extent from position 8
through position 36

Using Extents to Store Fields

- Store term positions in term inverted lists
- Define an extent inverted list for each field
- Include the document number and range of positions the extent includes

$$t_i \rightarrow (df_i, (d_1, tf_{i1}, (p_1, p_2, \dots)), \dots)$$

$$f_k \rightarrow ((d_1, (p_{start}, p_{end})), (d_2, (p_{start}, p_{end})), \dots)$$

fish	1,2	1,4	2,7	2,18	2,23	3,2	3,6	4,3	4,13
title	1:(1,3)	2:(1,5)							4:(9,15)

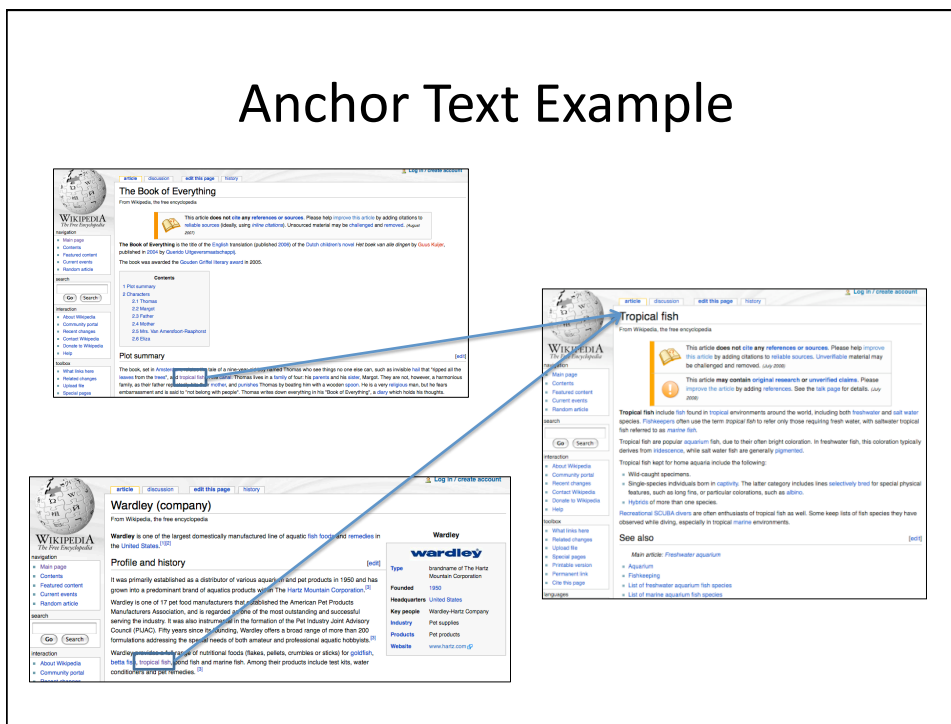
Field Storage Tradeoffs

- Include field info in inverted lists:
 - Storage efficient, fairly inflexible, fairly slow processing
- New lists for terms in fields:
 - Storage inefficient, more flexible, faster processing
- Field extents:
 - Storage efficient, very flexible, fairly fast processing

Anchor Text

- *Anchor text* is text on another page used to link to a document
- Can indicate what other people think the document is about
- Can be taken as a short summary of the documents contents

Anchor Text Example



Indexing Anchor Text

- Simple solution:
 - Include anchor text as part of document text
 - “Tropical” term frequency = # of times it appears in the document + # of times it appears in anchor text in documents linking to it
- Slightly more complex solution:
 - Include anchor text in fields, e.g. <anchor>
 - One field for each link to the document

Inverted Lists at Google

- As of 1998, Google stored the following:
 - Whether a term occurrence is “plain” or “fancy”
 - “Fancy” = occurs in URL, title, anchor text, or meta tag.
 - “Plain” = everything else
 - If plain, store:
 - Whether capitalized, font size information, and position information (in 1 bit, 3 bits, and 12 bits respectively)
 - If fancy, store:
 - Whether capitalized, maximum font size, type of hit, and position information (in 1 bit, 3 bits, 4 bits, and 8 bits respectively)
 - And if type=anchor, split 8 position bits into 4 docID bits and 4 position bits

Inverted Lists at Google

- Example: “tropical” occurs 3 times in document
 - Once capitalized in title at position 1
 - Once capitalized in a header at position 4
 - Once in lower-case in body text at position 108
- Also occurs in 2 other linking documents
- Google inverted list might look like this:

tropical	→	(df, (d ₁ , (1, 111, 0001, 00000001),	Fancy hit 1 (title)
		(1, 111, 0010, 00000100),	Fancy hit 2 (header)
		(0, 011, 000001101100),	Plain hit
•		(0, 111, 0100, 0110, 0100),	Anchor hit 1
		(0, 111, 0100, 0111, 0010))	Anchor hit 2