COMP90042
Web search and text analysis

Workshop Review

xudong.han@unimelb.edu.au
https://github.com/HanXudong/COMP90042_Workshops

```
from nltk.tokenize import word_tokenize
```

```
sentence = "Hello Aswathi How are you doing today"
sentence_token = word_tokenize(sentence)
sentence_token
```

```
['Hello', 'Aswathi', 'How', 'are', 'you', 'doing', 'today']
```

Tokenization

http://blog.xnextcon.com/?p=233

```python
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
```

```python
word = "mulitplying"
stem.stem(word)
```

```
'mulitpli'
```

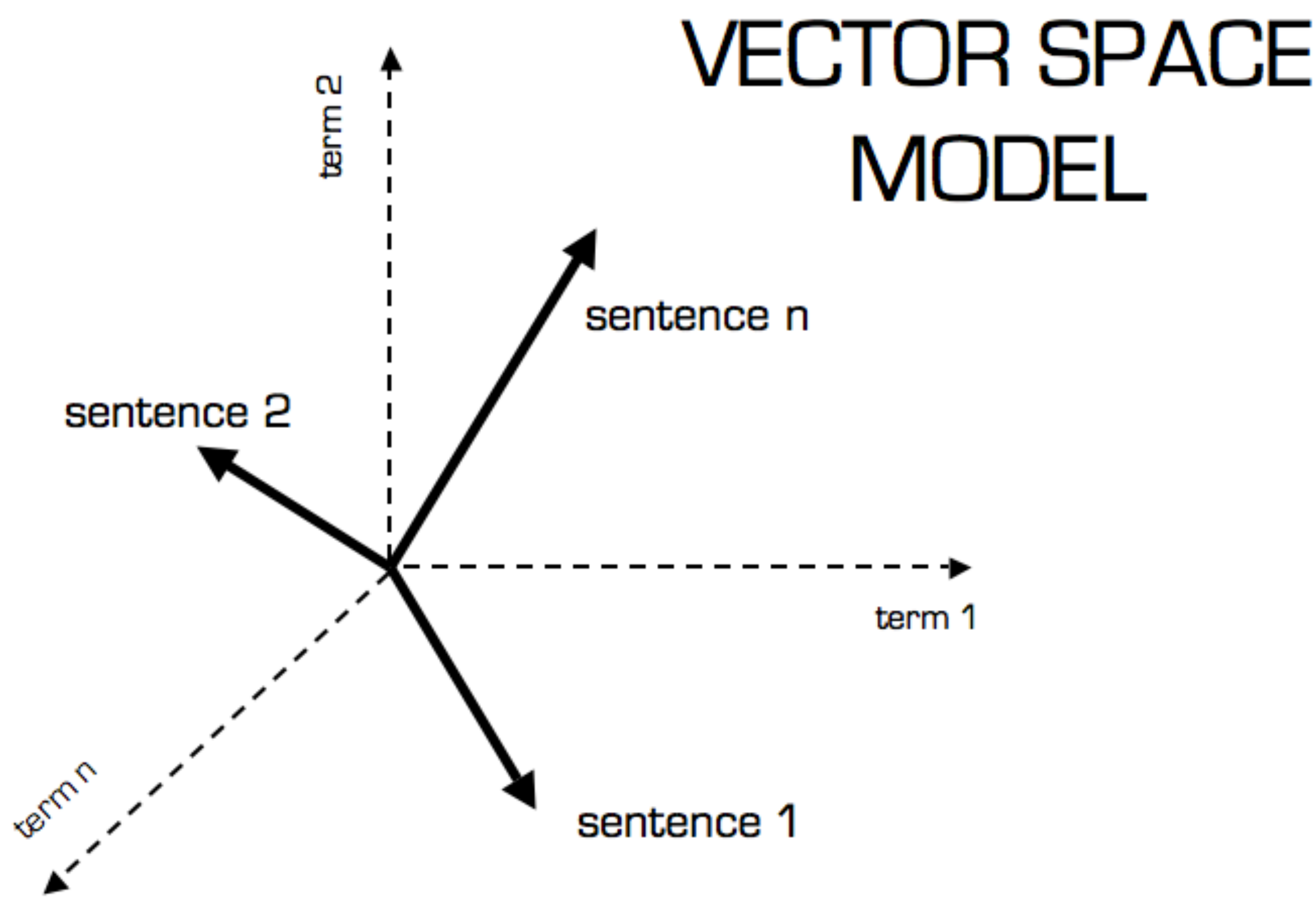
```python
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
```

```python
word = "multiplying"
lem.lemmatize(word,"v")
```

```
'multiply'
```

Stemming and Lemmatisation

http://blog.xnextcon.com/?p=233



Vector Space Model

http://bitsearch.blogspot.com/2011/01/vector-space-model-for-scoring.html

TDM & Inverted index

	two	tea	me	you
doc1	0.707	0.707	0	0
doc2	0	0.707	0.353	0.353
doc3	0	0	0.707	0.707

- Query 1: Tea me
- Query 2: Two

two	1: 0.707;
tea	1:0.707; 2: 0.707
me	2: 0.353; 3:0.707
you	2: 0.353; 3:0.707

TF*IDF & BM25

$$idf_t = log\frac{N}{df_t}$$

$$tf_{d,t} \times idf_t$$

$tf_{d,t}$: term frequency of a document (count of a term t in a document d)

idf_t : inverse document frequency

df_t : document frequency (count of documents that contain the term t)

$$w_t = log\frac{N - df_t + 0.5}{df_t + 0.5} \times \frac{(K_1 + 1)tf_{d,t}}{k_1((1 - b) + b\frac{L_d}{L_{avg}}) + tf_{d,t}} \times \frac{(k_3 + 1)tf_{q,t}}{k_3 + tf_{q,t}}$$

where $0 \le K_1 \le \infty,\ 0 \le K_3 \le \infty,$ **and** $0 \le b \le 1$

Posting List Compression

the	ids:	25	26	29	…	12345	12347
	gaps:	25	1	3	…	1	2
house	ids:	5213	5234	5454	5591	…	
	gaps:	5213	1	220	137	…	
aeronaut	ids:	251235	251239	251240			
	gaps:	251235	4	1			

Gaps between ids or term frequencies?

Variable Byte Compression

Encoding

```
function ENCODE(x)
    while x >= 128 do
        WRITE(x mod 128)
        x = x ÷ 128
    end while
    WRITE(x + 128)
end function
```

Decoding

```
function DECODE(bytes)
    x = 0, s = 0
    y = READBYTE(bytes)
    while y < 128 do
        x = x ^ (y << s)
        s = s + 7
        y = READBYTE(bytes)
    end while
    x = x ^ ((y − 128) << s)
    return x
end function
```

Q: why do we use “ ^ ”?

WAND

- **Top K retrieval**
- **Overestimate**

Query Q: The quick brown fox with $k = 2$

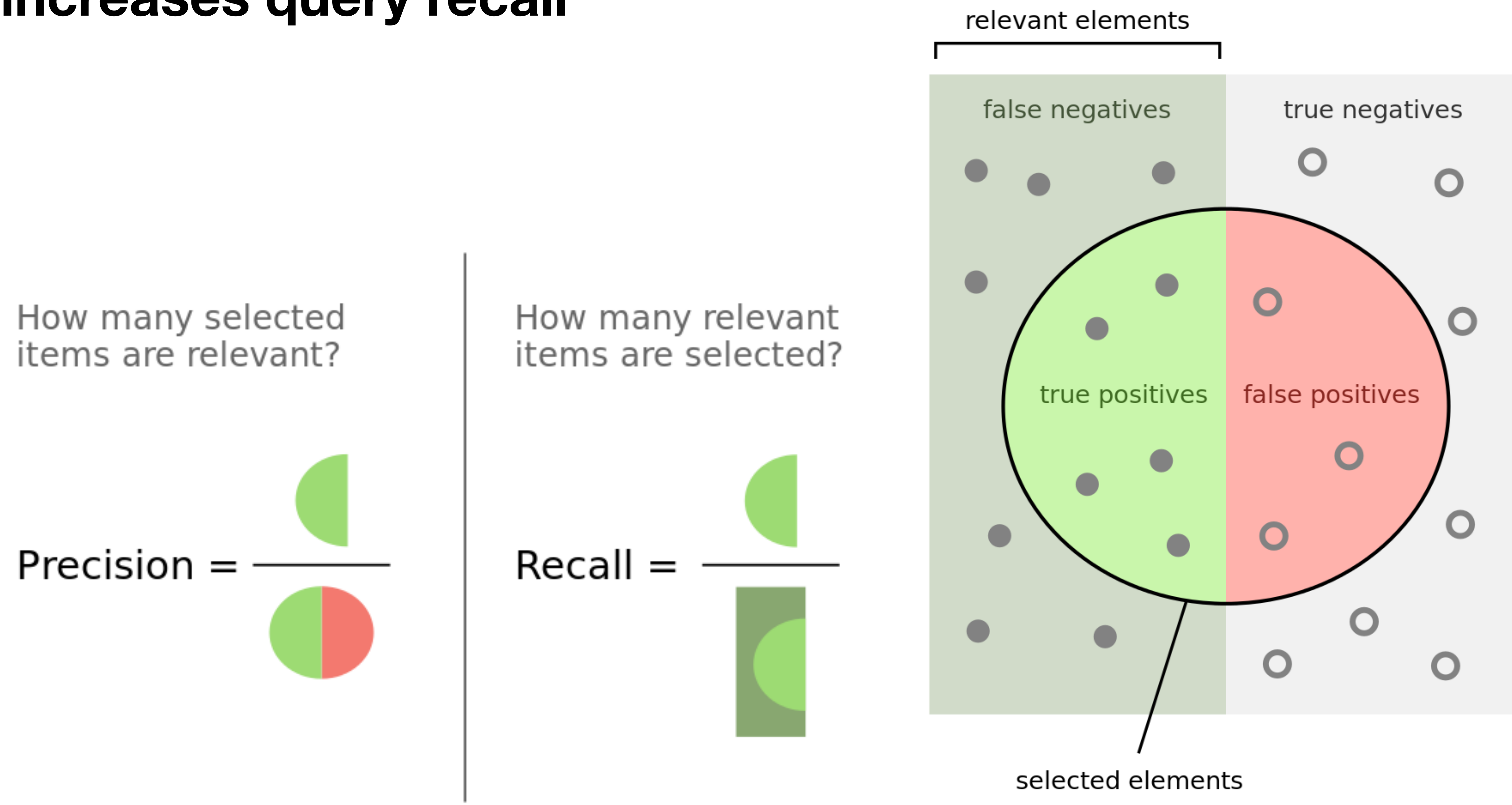
Maximum Contribution for each query term

	Max													
The	0.9	2	3	7	8	9	10	11	12	13	17	18	19	...
quick	1.9	5	6	9	11	14	18							
brown	2.3	2	4	5	15	42	84	96						
fox	7.1	5	7	8	13									

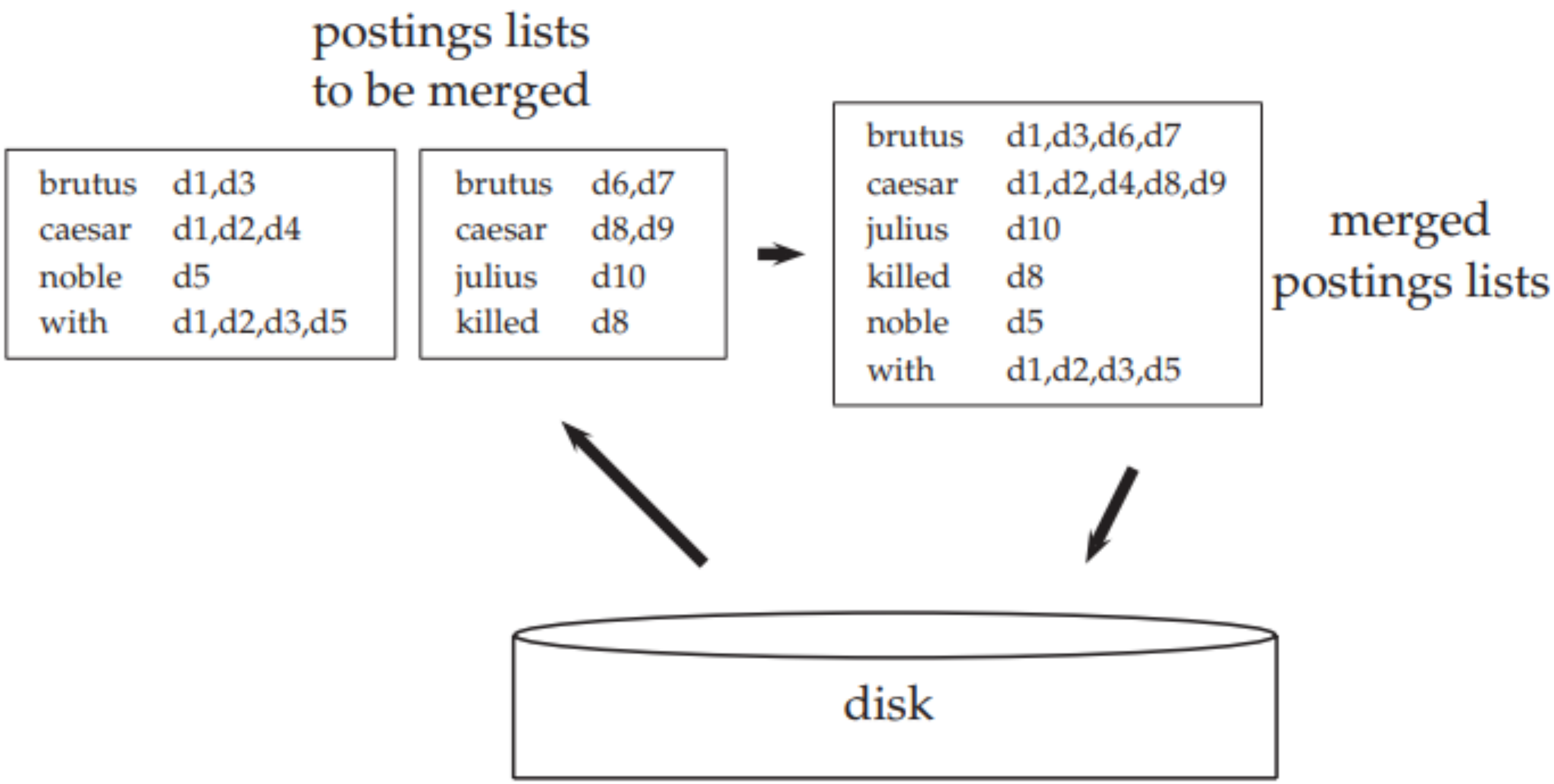
$$S_{TF-IDF}(d, Q) = \sum_{t \in Q} tf_{d,t} \times log \frac{N}{df_t}$$

Query Expansion Evaluation

Query expansion increases query recall



static inverted index construction and incremental index construction.



► **Figure 4.3** Merging in blocked sort-based indexing. Two blocks ("postings lists to be merged") are loaded from disk into memory, merged in memory ("merged postings lists") and written back to disk. We show terms instead of termIDs for better readability.

Why is a logarithmic index layout useful? What are the disadvantages of such an index structure?

- what is a logarithmic index layout?
  Use a logarithmic number($\log N$) of indexes. At each level i, store index of size $2^i \times n$

  http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html

- Query all logN indexes at the same time and merge results

what are the strengths and weaknesses of the methods above for evaluating IR systems?

$$Precision@k = \frac{\sum_{i=1}^{k} relevance_i}{k}$$

Precision@k

- Easy to evaluate and understand
- But no differentiation by rank for ranked document 1, 2, …, k
- But no adjustment for the size of the relevant documents

$$AP = \frac{\sum_{k=1}^{n} precision@k \times relevance_k}{\sum_{k=1}^{n} relevanc_k}$$

Average precision

- Differentiation by rank
- Adjustment for the size of the relevant documents
- But need to know the size of the relevant documents

$$RBP = (1-p) \times \sum_{i=1}^{n} r_i \times p^{i-1}$$

Rank biased precision

- Differentiation by rank
- Adjustment for the size of the relevant documents
- But need to decide on the persistence probability *p*

N-gram Model

1. how much wood would a wood chuck chuck if a wood chuck would chuck wood

2. a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

W_1,W_2	<s>a	<s>wood	chuck</s>
Count(w_1,w_2)	1	0	0
Count(w_1)	2	2	9
P(w_2|w_1)	1/2	0	0

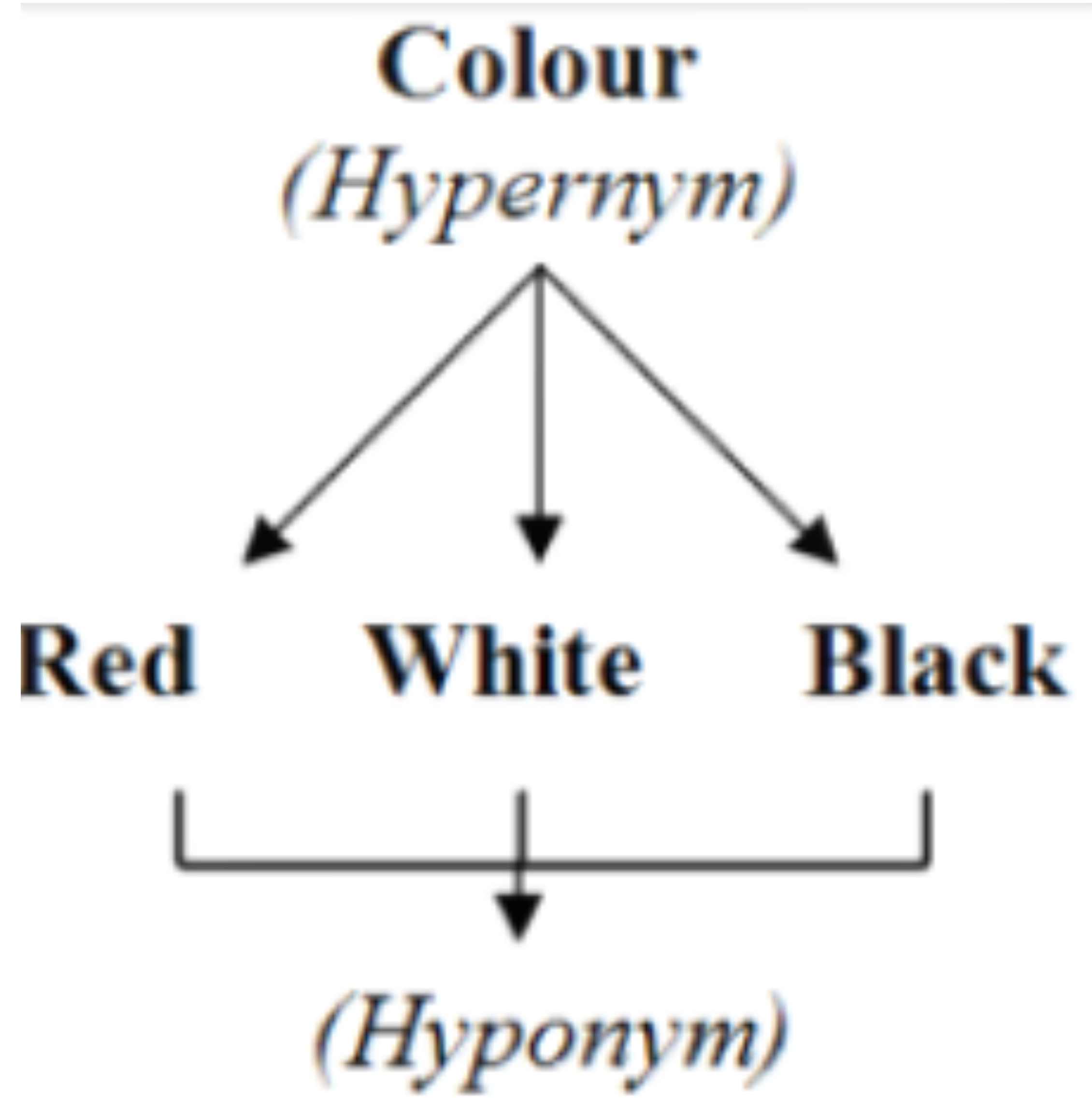
$$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-1})$$

- A: a wood could chuck;

- B: wood would a chuck ;

Smoothing, back–off and interpolation

- Add-one smoothing
- Add-k smoothing
- The idea in a Backoff model is to build an Ngram model based on an (N-1) model
- https://en.wikipedia.org/wiki/Katz%27s_back-off_model
- Interpolation: instead of just backing off to the non-zero Ngram, it is possible to take into account all Ngrams.
- Estimate lambdas from held-out dataset.



- **Meronym:** Part of a whole

- **Holonym:** The whole to which parts belong

		entity		
		abstraction...		
		communication		
		message...		entity
entity	entity	statement	entity	abstraction...
abstraction...	abstraction...	pleading	abstraction...	measure
communication	psychological...	charge...	group...	system of meas...
message...	cognition...	accusation...	collection...	information meas...

information

entity	entity	
physical...	abstraction...	
process...	psychological...	entity
processing	cognition...	abstraction...
data process...	process...	psychological...
operation	basic cog...	event
computer op...	memory...	act...

retrieval

information is more similar to the word *retrieval* or the word *science*

$$WuP\_sim(c_1, c_2) = \frac{2 \times depth(LCS(c_1, c_2))}{depth(c_1) + depth(c_2)}$$

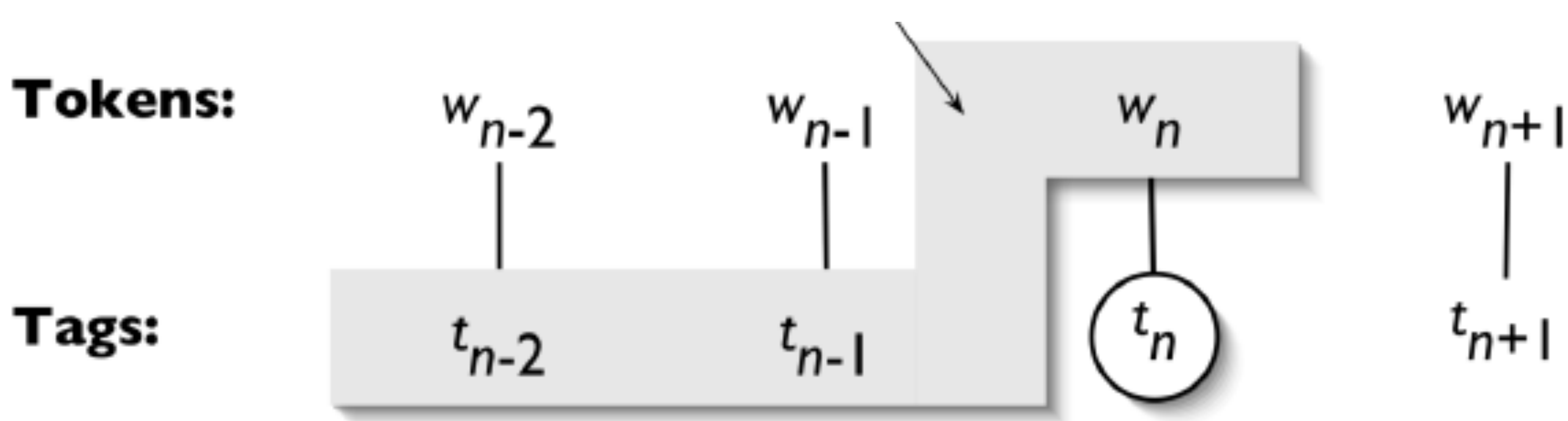
PMI

	cup	not (cup)	Total
world	55	225	**280**
not (world)	315	1405	**1720**
Total	**370**	**1630**	**2000**

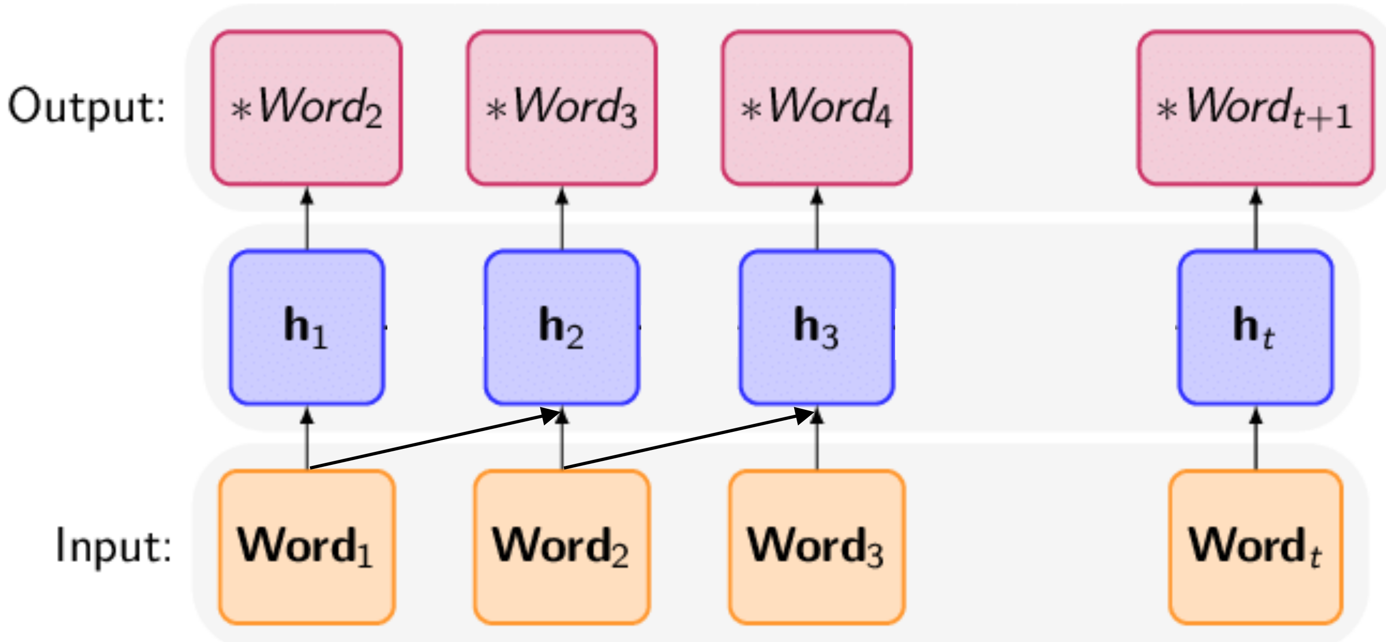
$$PMI(x,y) = log_2 \frac{p(x,y)}{p(x)p(y)} = log_2 P(x,y) - log_2 p(x) - log_2 P(y)$$

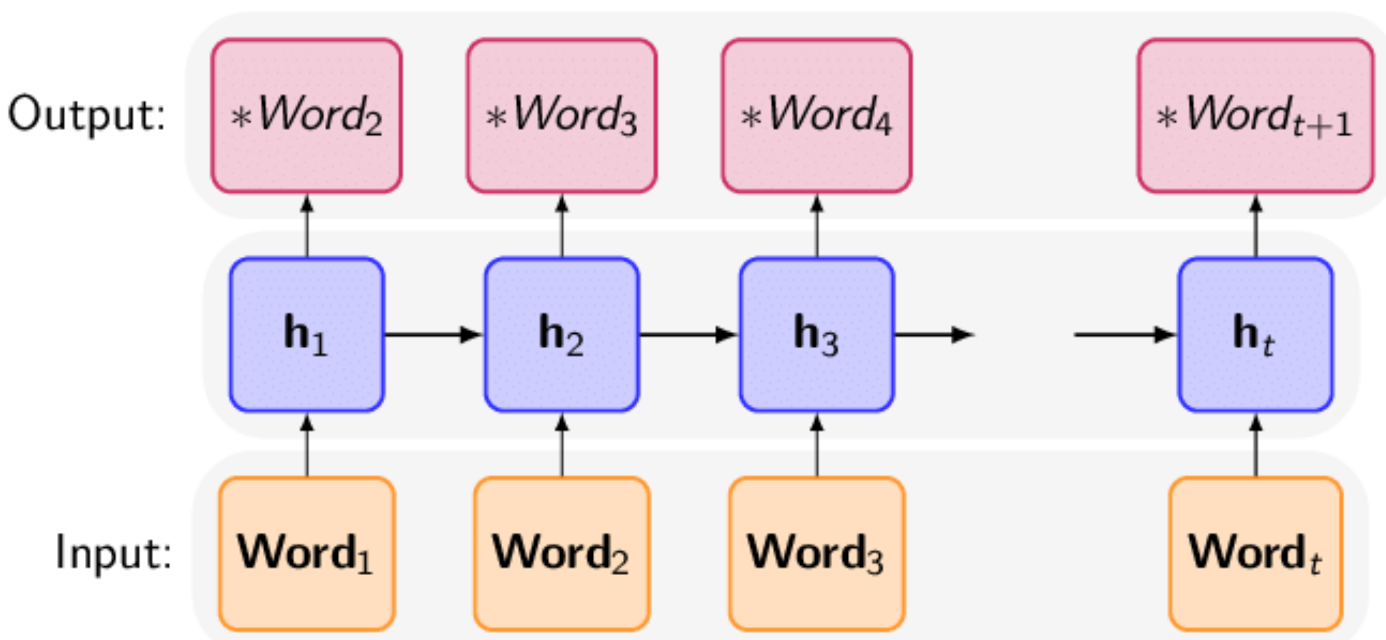
A part of speech (abbreviated form: PoS or POS) is a category of words which have similar grammatical properties.

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*’s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential ‘there’	*there*	PRP$	possess. pronoun	*your, one’s*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -



recurrent neural network (RNN) language model
feed-forward language model





NER & IOB

- IO V.S. IOB

- Apple is looking at buying U.K. startup for $1 billion

Apple	ORG
U.K.	GPE
$1 billion	MONEY

What is Relation Extraction? How is it similar to NER, and how is it different?

- Relation Extraction attempts to find and list the relationships between important events or entities within a document.

- Methods:
  - Rule-based
  - Supervised learning
  - Semi-supervised
  - Distant model
  - Unsupervised

  - E.g. Morgan's father is Tony. Tony's wife is pepper.

QA system
In a Relation Extraction sense:

– Offline, we process our document collection to generate a list of relations (our knowledge base)

– When we receive a (textual) query, we transform it into the same structural representation, with some known field(s) and some missing field(s)

– We examine our knowledge base for facts that match the known fields

– We rephrase the query as an answer with the missing field(s) filled in from the matching facts from the knowledge base

QA system
In an Information Retrieval sense:

– Offline, we process our document collection into a suitable format for IR querying (e.g. inverted index)

– When we receive a (textual) query, we remove irrelevant terms, and (possibly) expand the query with related terms

– We select the best document(s) from the collection based on our querying model (e.g. TF-IDF with cosine similarity)

– We identify one or more snippets from the best document(s) that match the query terms, to form an answer

α	*1:silver*	*2:wheels*	*3:turn*	
JJ:	0.24	0.0096	JJ $\to$ JJ	A[JJ,JJ]B[JJ, turn]
		JJ $\to$ JJ	0.0096	$\times 0.4 \times 0.1 = 0.000384$
			NNS $\to$ JJ	A[NNS,JJ]B[JJ, turn]
			0.048	$\times 0.1 \times 0.1 = 0.00048$
			VBP $\to$ JJ	A[VBP,JJ]B[JJ, turn]
			0.018	$\times 0.4 \times 0.1 = \mathbf{0.00072}$
NNS:	0.12	0.048	JJ $\to$ NNS	A[JJ,NNS]B[NNS, turn]
		JJ $\to$ NNS	0.0096	$\times 0.5 \times 0.3 = 0.00144$
			NNS $\to$ NNS	A[NNS,NNS]B[NNS, turn]
			0.048	$\times 0.4 \times 0.3 = \mathbf{0.00576}$
			VBP $\to$ NNS	A[VBP,NNS]B[NNS, turn]
			0.018	$\times 0.5 \times 0.3 = 0.0027$
VBP:	0.03	0.018	JJ $\to$ VBP	A[JJ,VBP]B[VBP, turn]
		NNS $\to$ VBP	0.0096	$\times 0.1 \times 0.6 = 0.000576$
			NNS $\to$ VBP	A[NNS,VBP]B[VBP, turn]
			0.048	$\times 0.5 \times 0.6 = \mathbf{0.0144}$
			VBP $\to$ VBP	A[VBP,VBP]B[VBP, turn]
			0.018	$\times 0.1 \times 0.6 = 0.00108$

Regular grammar & Regular language

- A language is a set of acceptable strings and a grammar is a generative description of a language.
- Regular language is a formal language that can be expressed using a regular expression.
- Regular grammar is a formal grammar defined by a set of production rules in the form of A -> xB, A - x and A->ϵ, where A and B are non-terminals, X is a terminal and ϵ is the empty string.
- A language is regular if and only if it can be generated by a regular grammar.

CFG & CYK parsing

0 1 2 3 4

a	man	saw	John
[0,1]	[0,2]	[0,3]	[0,4]
	[1,2]	[1,3]	[1,4]
		[2,3]	[2,4]
			[3,4]

Chomsky Normal Form (CNF)

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 12.5 The CKY algorithm.

Dependency parsing

Buffer	Stack	Action
Yesterday, I shot an elephant in my pyjamas.		Shift
I shot an elephant in my pyjamas	Yesterday	Shift
shot an elephant in my pyjamas	Yesterday, I	Shift
an elephant in my pyjamas	Yesterday, I, shot	Arc-Left (I <- shot)
an elephant in my pyjamas	Yesterday, shot	Arc-Left (Yesterday <- shot)

Universal dependencies

```
nmod:tmod(shot-4, Yesterday-1)
nsubj(shot-4, I-3)
root(ROOT-0, shot-4)
det(elephant-6, an-5)
dobj(shot-4, elephant-6)
case(pyjamas-9, in-7)
nmod:poss(pyjamas-9, my-8)
nmod(shot-4, pyjamas-9)
```

- two types of transitions
  - shift = move word from buffer on to top of stack
  - arc = add arc (left/ right) between top two items on stack and remove dependent from stack

Anaphors

- **Anaphor**: linguistic expressions that refer back to earlier elements in the text
- Anaphors have a **antecedent** in the discourse, often but not always a noun phrase

Yesterday, Ted was late for work. It all started when his car wouldn't start.

- Pronouns are the most common anaphor
- But there are various others
  - Demonstratives (*that problem*)
  - Definites (*the problem*)

Machine Translation

- Representation:

  $E = e_1 \ldots e_I =$ *And the program has been implemented*

  $F = f_1 \ldots f_J =$ *Le programme a ete mis en application*

  $A = a_1 \ldots a_J =$ 2, 3, 4, 5, 6, 6, 6.

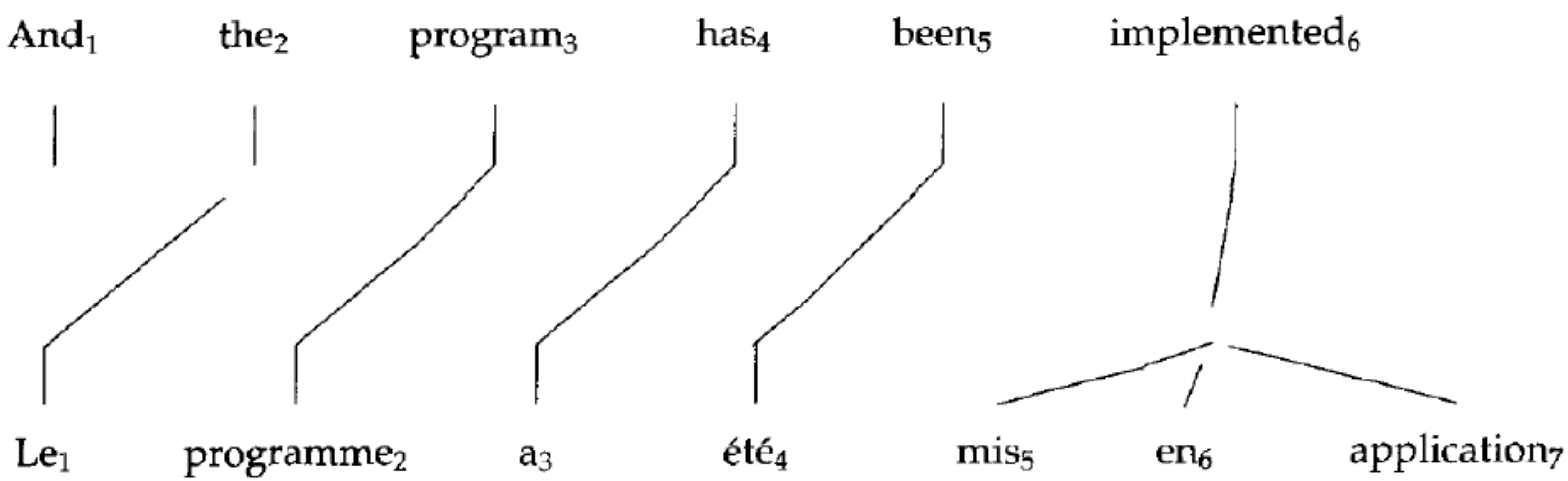


Figure from Brown, Della Pietra, Della Pietra, Mercer, 1993

Tips

- Slides & recording

- Workshops