Text Processing

h1

Text Processing - Lecture

> The content of this blog is based on the course COM6115(Text Processing (AUTUMN 2019~20)) at the University of Sheffield.
>
> @All the konowledge rights are reserved by the owner of this course's materials,Dr Mark Hepple and Dr Chenghua Lin at the University of Sheffield.

[toc]

h3

1. IR(Information Retrieval)

Information Retrieval (IR): concerned with developing algorithms and models for retrieving relevent documents from text collections.

- Text collection = some set of 'documents' / Query: user indication of what s/he want。
- Purpose: Finding pages that contain the words in the query by "relevance" to the query and clever indexing.

h4

Issues:

- How can I formulate a query?

  Normally keywords, could be natural language.

- How are the documents represented?

  indexing

- How does the system find the best-matching document?

retrieval model

- How are the results presented to me?

  unsorted list, ranked list, clusters

- How do we know whether the system is any good?

  evalution

> Indexing: h4

The task of finding terms that describe documents well

- Manual: indexing by humans (using fixed vocabularies) / labour and training intensive
- Automatic: Term manipulation (certain words count as the same term) / Term weighting (certain terms are more important than others) / Index terms must only derive from text

> MeSH — Medical Subject Headings (Manuall indexing)
>
> a very large controlled vocabulary for describing/indexing medical documents, e.g. journal papers and books. It provides a hierarchy of descriptors (a.k.a. subject headings).
>
> hierarchy has a number of top-level categories, e.g.: Anatomy [A] Organisms [B] Diseases [C] Chemicals and Drugs [D] Analytical, Diagnostic and Therapeutic Techniques and Equipment [E] Psychiatry and Psychology [F] Biological Sciences [G]
>
> Evalution:
>
> - Advantage: 1)High precision searches 2)Works well for closed collections (books in a library).
> - Disadvantage: 1)Searchers need to know terms to achieve high precision. 2)Labellers need to be trained to achieve consistency. 3)Collections are dynamic → schemes change constantly.

> Automatic Indexing:
>
> - Use the natural language as indexing language
> - Implementation: inverted files
>   - record each term, the ids of thedocuments in which it appears
>   - only matters if it does or does not appear - not how many times
>   - (doc_id)

Doc	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

⟹

Num	Token	Docs
1	cold	1, 4
2	days	3, 6
3	hot	1, 4
4	in	2, 5
5	it	4, 5
6	like	4, 5
7	nine	3, 6
8	old	3, 6
9	pease	1, 2
10	porridge	1, 2
11	pot	2, 5
12	some	4, 5
13	the	2, 5

Top
Foot

- also recoed the count of occurrences within each document.
- help find documents more relevant to query.
- (doc_id : occurance_number)

Doc	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

⟹

Num	Token	Docs
1	cold	1:1, 4:1
2	days	3:1, 6:1
3	hot	1:1, 4:1
4	in	2:1, 5:1
5	it	4:2, 5:1
6	like	4:2, 5:1
7	nine	3:1, 6:1
8	old	3:1, 6:1
9	pease	1:2, 2:1
10	porridge	1:2, 2:1
11	pot	2:1, 5:1
12	some	4:2, 5:1
13	the	2:1, 5:1

Automated retrieval models: h3

Bag-of-words Approach h4

- Standard approach to represent the documents
  - record what words are present
  - Usually, plus count of term in each document
- Ignore the relation between words
  - order of the words / permutation of the words

Terms h4

Common to just use the words, but pre-process them for generalisation.

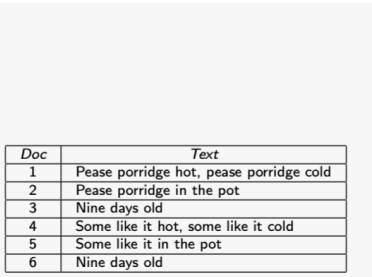
- Method:
  - Tokenisation: split words from punctuation (get rid of punctuation)
    - e.g. word-based. → word based three-issues: → three issues
  - Capitalisation: normalise all words to lower (or upper) case
    - e.g. Cat and cat should be seen as the same term, but should we conflate Turkey and turkey
  - Lemmatisation: conflate different inflected forms of a word to their basic form (singular, present tense, 1st person):
    - e.g. cats, cat → cat have, has, had → have worried, worries → worry
  - Stemming:
    - conflate morphological variants by chopping their affix
  - Normalisation: heuristics to conflate variants due to spelling, hyphenation, spaces, etc.

Top
Foot

- e.g. USA and U.S.A. and U S A → USA

  e.g. chequebook and cheque book → cheque book

  e.g. word-sense and word sense → word-sense

- Problem of Single and Multi-words
  - To aid recognition of phrases, might allow multi-word terms
  - Solution
    - identify multi-word phrases during retrieval
    - Positional indexes, storing position terms in documents, can help
- Positional indexs:
  - 

Doc	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

Num	Token	Docs
1	cold	1:(6), 4:(8)
2	days	3:(2), 6:(2)
3	hot	1:(3), 4:(4)
4	in	2:(3), 5:(4)
5	it	4:(3, 7), 5:(3)
6	like	4:(2, 6), 5:(2)
7	nine	3:(1), 6:(1)
8	old	3:(3), 6:(3)
9	pease	1:(1, 4), 2:(1)
10	porridge	1:(2, 5), 2:(2)
11	pot	2:(5), 5:(6)
12	some	4:(1, 5), 5:(1)
13	the	2:(4), 5:(5)

  - Approach:
    - Binary weights - 0/1: whether or not term is present in document
      - But documents with multiple occurrences of query keyword may be more relevant
    - Frequency of term in document: like the examples we have seen
      - Common terms: not very useful for discriminating relevant documents
    - Frequency in document vs in collection: weight terms highly if
      - They are frequent in relevant documents . . . but
      - They are infrequent in collection as a whole

Stop List

h4

- Use Stop list removal to exclude “non-content” words
- Usually most frequent (and least useful for retrieval)
- Effect:
  - greatly reduces the size of the inverted index

IR Method

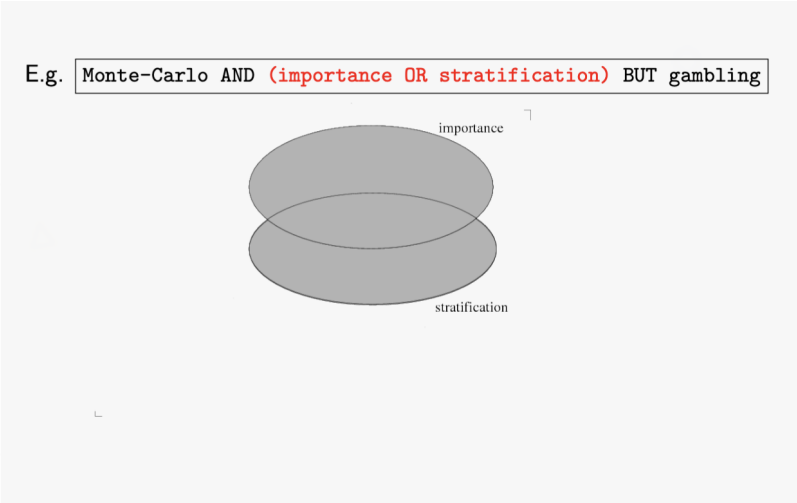
h4

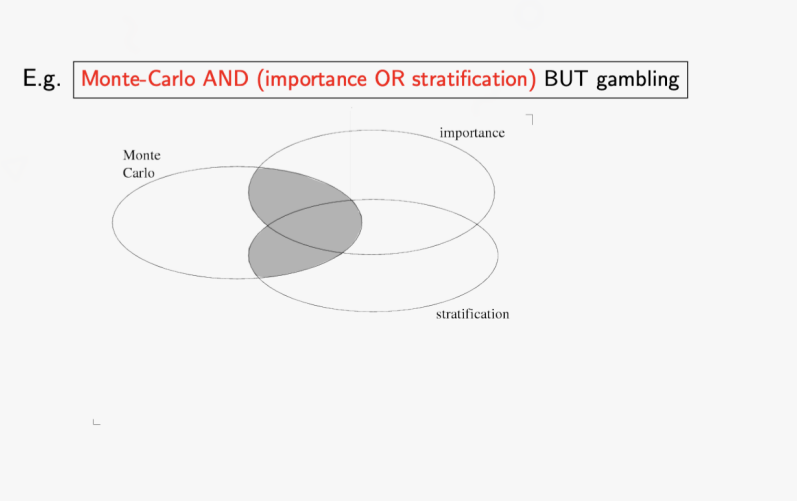
> Boolean Method(Binary)
>
> - Purpose:
>   - evaluate the relevance and suffience of term for match
>   - provides a simple logical basis for deciding whether any document should be returned.
>   - whether basic terms of query do/do not appear in the document

Top

Foot

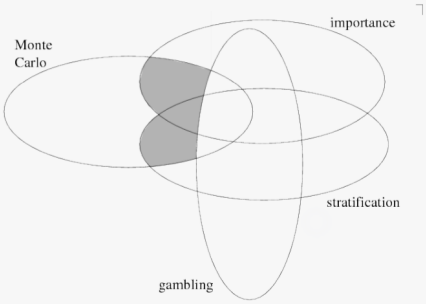
    - the meaning of the logical operators

- Algorithm:
    - frequency of document terms
    - not all search terms necessarily present in document
    - vector space model
- Approach:
    - Basic search terms(keywords)
    - using boolean operators
- Boolean Operators:
    - AND, OR, NOT, BUT, XOR (exclusive OR)
    - E.g.: Monte-Carlo AND (importance OR stratification) BUT gambling
- Set-theoretic interpretation:
    - $d(E)$ denote the document set for expression E
    - E either a basic term or boolean expression
        - $d(E1\ AND\ E2) = d(E1) \cap d(E2)$
        - $d(E1\ OR\ E2) = d(E1) \cup d(E2)$
        - $d(NOT\ E) = d(E)^c$
        - $d(E1\ BUT\ E2) = d(E1) - d(E2)$
    - Some example:
        - 

        - 

        -

E.g. Monte-Carlo AND (importance OR stratification) BUT gambling



- Conclusion:
  - Advantage:
    - Higher requirements for user: Expert knowledge needed to create high-precision queries
    - Often used by bibliographic search engines (library)
  - Disadvantage:
    - Most users not familiar with writing Boolean queries → not natural
    - Most users don't want to wade through 1000s unranked result lists → unless very specific search in small collections
    - This is particularly true of web search → large set of docs

Vector Space Model

- Description:
  - Documents are points in high-dimensional vector space
    - each term in index is a dimension → sparse vectors
    - values are frequencies of terms in documents, or variants of frequency
  - Queries are also represented as vectors (for terms that exist in index)
- Priori:
  - Select document(s) with highest document–query similarity
  - Document–query similarity is a model for relevance (ranking)
  - With ranking, the number of returned documents is less relevant → users start at the top and stop when satisfied
- Approach:
  - compare vector of query against vector of each document
    - to rank documents according to their similarity to the query
    -

Top
Foot

	$Term_1$	$Term_2$	$Term_3$	...	$Term_n$
Doc_1	9	0	1	...	0
Doc_2	0	1	0	...	10
Doc_3	0	1	0	...	2
...	...	...	...	...	...
Doc_N	4	7	0	...	5
Q	0	1	0	...	1

- Each document and query are represented as a vector of n values:

  $\overrightarrow{d^i} = (\overrightarrow{d_1^i}, \overrightarrow{d_2^i}, \overrightarrow{d_3^i}, \cdots, \overrightarrow{d_n^i})$ $\vec{q} = (q_1, q_2, q_3, \cdots, q_n)$

- Similarity: $\sqrt{\sum_{k=1}^{N}(q_k - d_k)^2}$
  - E.g.:
    - Doc1 and Q

      $\sqrt{(9-0)^2 + (0-1)^2 + (1-0)^2 + (0-1)^2} = 9.15$

    - Doc2 and Q

      $\sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (10-1)^2} = 9.15$

- Evalution:
  - distance is large for vectors of different lengths, even if by only one term (e.g. Doc2 and Q)
  - means frequency of terms given too much impact
  - Better similarity metric, used in vector-space model: cosine of the angle between two vectors $\vec{x}$ and $\vec{y}$: (For the each value of text and query, when the term occurs, the value should be 1 and if not, the value should be 0 for each x_i and y_i)

    $cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x^2} \sqrt{\sum_{i=1}^{n} y^2}}$

    - It can be interpreted as the normalised correlation coefficient:

      i.e. it computes how well the x_i and y_i correlate, and then divides by the length of the vectors, to scale for their magnitude.

    - Value:
      - range from:
        - 1, for vectors pointing in the same direction, to
        - 0, for orthogonal vectors,
        - -1, for vectors pointing in opposite directions

Inverse document frequency(**IDF**)

Term Weighting

Top
Foot

document collection	D	collection (set) of documents
size of collection	$\|D\|$	total number of documents in collection
term freq	$tf_{w,d}$	number of times w occurs in document d
collection freq	cf_w	number of times w occurs in collection
document freq	df_w	number of documents containing w

- informativeness of terms
- Idea that less common terms are more useful to finding relevant docs
- document frequency(df) reflects this difference
- collection frequency(cf) fails to distinguish them (i.e. very similar counts)
- Informativeness is inversely related to (document) frequency
- less common terms are more useful to finding relevant documents
- more common terms are less useful to finding relevant documents
- Compute

$$\frac{|D|}{df_w}$$

- Value reduces as df_w gets larger, tending to 1 as df_w approaches $ | D | $
- Value very large for small dfw— over-weights such cases
- To moderate this, take **log**: Inverse document frequency(**IDF**) $idf_{w,D} = log\frac{|D|}{df_w}$
  - Consider following counts (from New York Times data, $|D| = 10000$):

Word	cf_w	df_w
insurance	10440	3997
try	10422	8760

  - $log\frac{10000}{3997} = 0.398$ (insurance) $log\frac{10000}{8760} = 0.057$ (try) $log\frac{10000}{350} = 1.456$ (mischief)
- BUT Not all terms describe a document equally well
  - Putting it all together: **tf.idf**
    - Terms which are frequent in a document are better: $\mathbf{tf_{w,d} = freq_{w,d}}$
    - Combine the two to give tf.idf term weighting: $\mathbf{tf.idf_{w,d,D} = tf_{w,d} \cdot idf_{w,D}}$
    - e.g.

Top
Foot

tf.idf example:

Term	tf	df	$\|D\|$	idf	$tf.idf$
the	312	28,799	30,000	0.018	5.54
in	179	26,452	30,000	0.055	9.78
general	136	179	30,000	2.224	302.50
fact	131	231	30,000	2.114	276.87
explosives	63	98	30,000	2.486	156.61
nations	45	142	30,000	2.325	104.62
haven	37	227	30,000	2.121	78.48

For term the:

$$idf(the) = \log_{10}(\frac{30,000}{28,799}) = 0.018$$

$$tf.idf(the) = 312 \cdot 0.018 = 5.54$$

- e.g.

Example: Vector Space Model, tf.idf term weighting, cosine similarity

- tf.idf values for words in two documents D_1 and D_2, and in a query Q "hunter gatherer Scandinavia":

	Q	D_1	D_2	
hunter	19.2	56.4	112.2	
gatherer	34.5	122.4	0	
Scandinavia	13.9	0	30.9	
30,000	0	457.2	0	
years	0	12.4	0	
BC	0	200.2	0	
prehistoric	0	45.3	0	
deer	0	0	23.6	
rifle	0	0	452.2	
Mesolithic	0	344.2	0	
$\sqrt{\sum_{i=1}^{n} x_i^2}$	41.9	622.9	467.5	(i.e. length of vector)

- $$sim(\vec{q}, \vec{d}) = cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

- $$sim(\vec{q}, \vec{d}) = cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

$$\begin{aligned} cos(Q, D_1) &= \frac{(19.2 * 56.4) + (34.5 * 122.4) + \cdots + (0 * 0) + (0 * 344.2)}{41.9 * 622.9} \\ &= \frac{5305.68}{26071.72} \\ &= 0.20 \end{aligned}$$

$$\begin{aligned} cos(Q, D_2) &= \frac{(19.2 * 112.2) + (34.5 * 0) + \cdots + (0.0 * 452.2) + (0.0 * 0.0)}{41.9 * 467.5} \\ &= \frac{2583.8}{19570.0} \\ &= 0.13 \end{aligned}$$

- so document D_1 is more similar to Q than D_2

PageRank Algorithm

- Key method to exploit link structure of web: PageRank algorithm
  - Assigns a score to each page on web: its PageRank score
  - can be seen to represent the page's authority (or quality)
- Idea:
  - Link from page A to page B confers authority on B
  - how much authority is conferred depends on:
    - the authority (PageRank score) of A, and its number of out-going links

Top
Foot

        - i.e. A's authority is shared out amongst its out-going links
    - note that this measure is recursively defined
        - i.e. score of any page depends on score of every other page
    - PageRank scores have an alternative interpretation:
        - probability that a random surfer will visit that page
    - its PageRank score: a measure of its authority

Summary

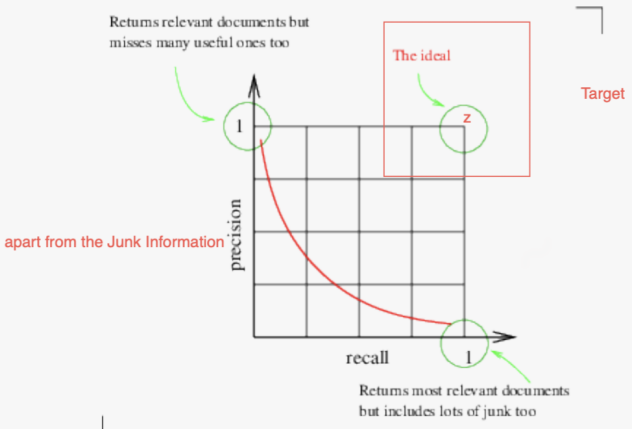
- Component / technique
    - Ranking (cosine, dot-product, . . .)
    - Term selection (stopword removal, stemming, . . .)
    - Term weighting (binary, TF, TF.IDF, . . .)
- Relevance
    - Evaluation of effectiveness in relation to the relevance of the documents retrieved
    - Relevance is judged in a binary way, even if it is in fact a continuous judgement
        - Impossible when the task is to rank thousands or millions of options: too subjective, too difficult
    - In IR research/development scenarios, one cannot afford humans looking at results of every system/variant of system
    - Instead, performance measured/compared using a pre-created benchmarking corpus, a.k.a. gold-standard dataset, which provides:
        - a standard set of documents, and queries
        - a list of documents judged relevant for each query, by human subjects
        - relevance scores, usually treated as binary
- Evaluation of IR systems – Metrics
    - AIM
        - get as much good stuff as possible
        - get as little junk as possible
    - The two aspects of this aim are addressed by two separate measures — recall and precision.

	Relevant	Non-relevant	Total
Retrieved	A	B	A+B
Not retrieved	C	D	C+D
Total	A+C	B+D	A+B+C+D

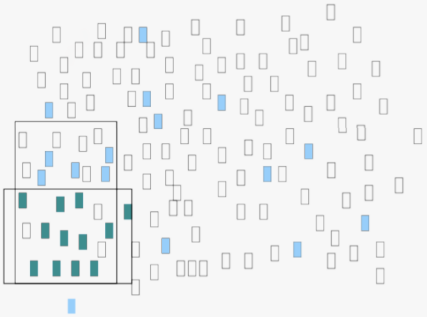
        - Recall : $\frac{A}{A+C}$ proportion of relevant documents returned
        - Precision: $\frac{A}{A+B}$ proportion of retrieved documents that are relevant
        - Both measures have range: [0 . . . 1]
        - Precision and Recall address the relation between the retrieved and relevant sets of documents
    - Trade-off Between Recall and Precision

Top
Foot

- There is always a trade-off between precision and recall
  - For IR: as more results are considered down the list, precision generally drops, while recall generally increases



- Recall and Precision - Which is the better system?



$$R_1 = \frac{A_1}{A_1+C_1} = \frac{16}{28} = .57$$
$$P_1 = \frac{A_1}{A_1+B_1} = \frac{16}{25} = .64$$

$$R_2 = \frac{A_2}{A_2+C_2} = \frac{12}{28} = .43$$
$$P_2 = \frac{A_2}{A_2+B_2} = \frac{12}{15} = .8$$

- Which did better: System 1 or System 2?

- F-measure
  - F measure (also called F1):
    - combines precision and recall into a single figure, and it is a harmonic mean
    - gives equal weight to both:

      P = Precision R = Retrievaled $\mathbf{F} = \frac{\mathbf{2PR}}{\mathbf{P+R}}$

    - penalises low performance in one value more than arithmethic mean:

	values	mean	F
e.g.	P=0.5, R=0.5	0.5	0.5
	P=0.1, R=0.9	0.5	0.18

  - F_β
    - allows user to determine relative importance of P vs. R, by varying β
    - F1 is a special case of Fβ (where β = 1)
- Precision at a cutoff

- Measures how well a method ranks relevant documents before non-relevant documents。

- E.g. there are **5 relevant documents = d1,d2,d3,d4,d5** – compute precision at top 5

	System 1		System 2		System 3	
	d1:	√	d10:	×	d6:	×
	d2:	√	d9:	×	d1:	√
	d3:	√	d8:	×	d2:	√
	d4:	√	d7:	×	d10:	×
rank 5:	d5:	√	d6:	×	d9:	×
	d6:	×	d1:	√	d3:	√
	d7:	×	d2:	√	d5:	√
	d8:	×	d3:	√	d4:	√
	d9:	×	d4:	√	d7:	×
rank 10:	d10:	×	d5:	√	d8:	×
precision at rank 5:		1.0		0.0		0.4
precision at rank 10:		0.5		0.5		0.5

  - Average Precision
    - Aggregates many precision numbers into one evaluation figure
    - Precision computed for each point a relevant document is found, and figures averaged

System 1			System 2			System 3		
d1:	√	(1/1)	d10:	×		d6:	×	
d2:	√	(2/2)	d9:	×		d1:	√	(1/2)
d3:	√	(3/3)	d8:	×		d2:	√	(2/3)
d4:	√	(4/4)	d7:	×		d10:	×	
d5:	√	(5/5)	d6:	×		d9:	×	
d6:	×		d1:	√	(1/6)	d3:	√	(3/6)
d7:	×		d2:	√	(2/7)	d5:	√	(4/7)
d8:	×		d3:	√	(3/8)	d4:	√	(5/8)
d9:	×		d4:	√	(4/9)	d7:	×	
d10:	×		d5:	√	(5/10)	d8:	×	
precision at rank 5:		1.0			0.0			0.4
precision at rank 10:		0.5			0.5			0.5
avg. prec:		1.0			0.354			0.573

Evaluation

h4

- Strengths
  - Can search huge document collections very rapidly
  - Insensitive to genere and domain of the texts
  - Relatively straightforward to implement
    - challenges scaling to huge, dynamic document collections
- Weakness:
  - Documents are returned not information/answers
    - user must further read texts to extract information
    - output is unstructured so limited possibilities for direct data mining/further processing

Top
Foot

2. Sentiment Analysis

h3

Abstract

- Explain the relevance of the topic
- Differentiate between objective and subjective texts
- List the main elements in a sentiment analysis system
- Provide a critical summary of the main approaches to the problem
- Explain how sentiment analysis systems are evaluated.

General Goal:

- Extract opinions, sentiments and emotions expressed by humans in texts and use this information for business, intelligence, etc. purposes. Can't be done manually: huge volumes of opinionated text (esp. Big Data on the Web).
  - Product review mining: Which features of the iPhone 11 customers like and which do they dislike?
  - Review classification: Is a movie review positive or negative?
  - Tracking sentiments toward topics over time: Is anger about the government policies growing or cooling down?
  - Prediction (election outcomes, market trends): Will the Tories win the next election?

Sentient Analysis

An **opinion** is a quintuple $(o_j, f_{jk}, so_{ijkl}, h_i, t_l)$, where:
- o_j is a target object.
- f_{jk} is a feature of the object o_j.
- so_{ijkl} is the sentiment value of the opinion of the
- opinion holder h_i (usually the author of the post)
- on feature f_{jk} of object o_j at time t_l.

so_{ijkl} is positive, negative, neutral, or a more granular rating, such as 1-5 stars as in movie reviews.

- The task of opinion mining is: given an opinionated document:
  - Discover all quintuples
  - Discover some of these components
- With that, one can structure the unstructured:
  - Traditional data and visualisation tools can be used to slice, dice and visualise the results.
  - Qualitative and quantitative analysis can be done.

Binary (lexicon-based)

h4

- Rule-based subjectivity classifier: a sentence/document is subjective.

  if it has at least n (say 2) words from the emotion words lexicon; a sentence/document is objective otherwise.

- Rule-based sentiment classifier: for subjective sentences/documents, count positive and negative words/phrases in the sentence/document.

  If more negative than positive words/phrases, then negative; otherwise, positive (if equal, neutral).

Top
Foot

- Rule-based sentiment classifier (feature-level):
  - Assume features can be identified in previous step by information extraction techniques, e.g., battery, phone, screen.
  - For each feature, count positive and negative emotion words/phrases from the lexicon.
  - If more negative than positive words/phrases, then negative; otherwise, positive (if equal, neutral).
- Rule-based sentiment classifier (feature-based)

  - Simple approach:
    - Input: a pair (f, s), where f is a product feature and s is a sentence that contains f.
    - Output: whether the emotion on f in s is positive, negative, or neutral.
    - **Step 1**: work on the sentence s containing f.
    - **Step 2**: select emotion words in s: $w_1, .., w_n$.
    - **Step 3**: assign orientations for these emotion words: 1 = positive, -1 = negative, 0 = neutral.
    - **Step 4**: sum up the orientation and assign the orientation to (f, s) accordingly.

Caveats

h4

- Other emotion words have context-dependent orientations, e.g.
  - small power consumption = positive
  - small screen = negative
- Can store more fine-grained sentiment information in lexicon and add additional rules.

Gradable

h4

> - Use of ranges of sentiment instead of a binary system, to deal with intensifiers like:
>   - absolutely, utterly, completely, totally, nearly, virtually, essentially, mainly, almost, e.g.: absolutely awful
> - And grading adverbs like:
>   - Very, little, dreadfully, extremely, fairly, hugely, immensely, intensely, rather, reasonably, slightly, unusually, e.g.: a little bit cold

- Rule-based gradable sentiment classifier
  - The lexicon: word-lists with pre-assigned emotional weights, e.g:

    Neg. dimension (C_neg): {-5,…,-1}, Pos. dimension (C_pos): {+1,…,+5}

Top
Foot

```
bore      -3      careful 3
boring    -3      careless    -2
bother    -1      cares   2
brave     3       caring  3
bright    2       casual  2
brillian      2   casually    2
broke     -1      certain 2
brutal    -3      challeng    2
burden    -1      champ   2
calm      2       charit  2
care      2       charm   2
cared     2       cheat   -3
carefree      2
```

- Additional general rules to change the original weights:
  - Negation rule:
    E.g.: “I am not good today”. Emotion(good)= +3;
    “not” is detected in neighbourhood (of 5 words around);
    so emotional valence of “good” is decreased by 1 and sign is inverted → Emotion(good) = −2
  - Capitalization rule:
    E.g. “I am GOOD today”.
    Emotion(good)= +3; Add +1 to positive words → Emotion(GOOD) = +4
    Likewise, in “I am AWFUL today”.
- **Intensifier rule**:
  - Needs a list of intensifiers: “definitely”, “very”, “extremely”, etc.
  - Each intensifiers has a weight
  - The weight is added to positive terms
  - The weight is subtracted from negative terms
  - E.g.: “I am feeling very good”.
    Emotion(good)= +3; emotional valence of “good” increased by 1
    → Emotion(good) = +4
  - E.g. “This was an extremely boring game”
    Emotion(boring)=−3; emotional valence of “boring” decreased by −2
    → Emotion(boring) = −5
- Diminisher rule:
  - Need a list: “somewhat”, “barely”, “rarely”, etc.
  - Each intensifiers has a weight
  - The weight is subtracted from positive terms
  - The weight is added to negative terms
  - E.g.: “I am somewhat good”. Emotion(good)= +3; emotional valence of “good” decreased by 1 → Emotion(good) = +2
  - E.g. “This was a slightly boring game” Emotion(boring)=−3; emotional valence of “boring” increased by 1 → Emotion(boring) = −2

Top
Foot

- **Exclamation rule**:
  - Functions like intensifiers.
  - E.g.: “Great show!!!”. Emotion(great)= +3; Weight(!!!) = 2 → Emotion(great) = 5
- Emoticon rule:
  - Each has its own emotional weight, like an emotion word.
  - E.g.: Emotion(🙂) = +2; Emotion(☹) = −2.

    E.g.: “I can’t believe this product 😠
- Decision
  - **Final decision based on ALL emotion words:**
    - If $|C_{pos}| > |C_{neg}|$ then {positive}
    - If $|C_{pos}| < |C_{neg}|$ then {negative}
    - If $|C_{pos}| = |C_{neg}|$ then {neutral}

    - E.g.: “He is brilliant but boring”: Emotion(brilliant) $= 2$; Emotion(boring) $= -3$ $\rightarrow C_{pos} = 2$, $C_{neg} = -3$, so {negative}
    - E.g.: “I am not good today”: Emotion(good) $= -2$ $\rightarrow C_{pos} = 0$, $C_{neg} = -2$, so {negative}
    - E.g.: “I am not GOOD today”: (Emotion(good)=3) → ??? Negative
    - E.g.: “I am so surprised by this product!!! ☹”: (Emotion(☹)=−2) → ??? Negative
- Evalution:
  - Advantages:
    - Works effectively with different texts: forums, blogs, etc.
    - Language independent - as long as an up-to-date lexicon of emotion words is available
    - Doesn’t require data for training
    - Can be extended with additional lexica, e.g. for new emotion words/symbols as they become popular, esp. in social media
  - Disadvantages:
    - Requires a lexicon of emotion words, which should be fairly comprehensive, covering new words, abbreviations (LOL, m8, etc.), misspelled words, etc.
- Collect relevant words/phrases that can be used to express sentiment. Determine the emotion of these subjective word/phrases.
  - Manually: word lists with pre-assigned emotional weights
  - Semi-automatically
    - Dictionary-based: find synonyms/antonyms of seed emotion words in dictionaries like WordNet
    - Corpus-based: find synonyms/antonyms of seed emotion words in corpora
  - Semi-automatically created from seed words: start with seed positive and negative words:
    - Search for synonyms/antonyms in dictionaries like WordNet; OR

Top
Foot

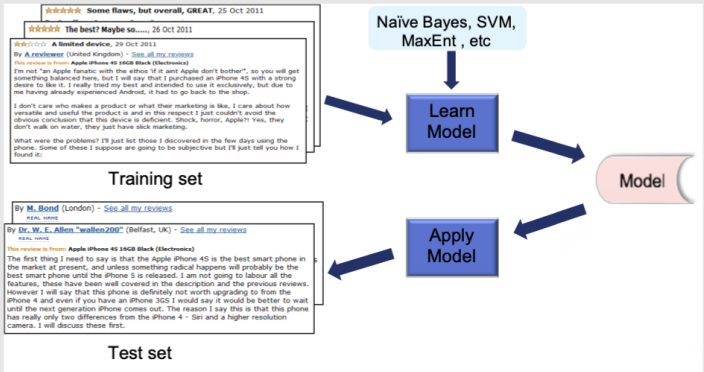
- Build patterns from seed words/phrases to search on large corpora, like the Web:
  - “beautiful and” (+)
  - “low cost but”(-)
  - “very nice and”(+)
- Machine Learning
  - Subjectivity classifier: first run binary classifier to identify and then eliminate objective segments
  - Sentiment classifier with remaining segments: learn how to combine and weight different attributes to make predictions. E.g. Naive Bayes

Corpus-based (Machine Learning)

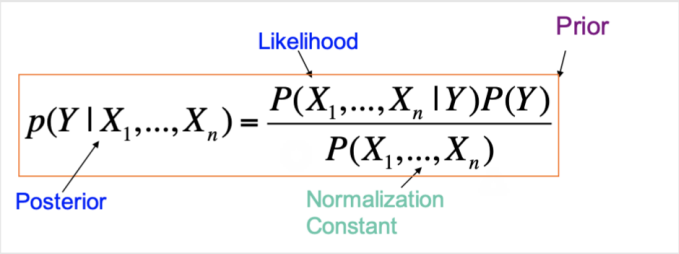
h4

Basic Knowledge:

- **Supervised learning:** the machine learning task of inferring a function from labeled training data
- Given:
  - **Target:** a fixed set of **classes**: $Y = y_1, y_2, \ldots, y_n$, e.g. {sports, politics, . . . , music}
  - **Training data:** a collection of data objects X with known classes Y, i.e. $(X, Y) = (x_1, y_1), (x_2, y_2) \ldots (x_n, , y_n)$. E.g {(d1, sports), (d2, sports), (d3, music) . . . }.
  - **Testing data:** a description of an unseen instance, Dnew e.g. a new document **without** class label information
- Goal:
  - Predict the category/class of D_{new} : $y(x) \in Y$, where $y(x)$ is a *classification function*, aka *trained model*, whose domain is X and whose range is Y.



- Rely on syntactic or co-occurrence patterns in large text corpora



- $P(Y)$: Prior belief (probability of hypothesis Y before seeing any data)
- $P(X_1, ..., X_n|Y)$: Likelihood (probability of the data if the hypothesis Y is true)
- $P(X_1, ..., X_n)$: Data evidence (marginal probability of data)
- $P(Y|X_1, ..., X_n)$: Posterior (probability of hypothesis Y after having seen the data)

- Assume A and B are Boolean Random variables. Then

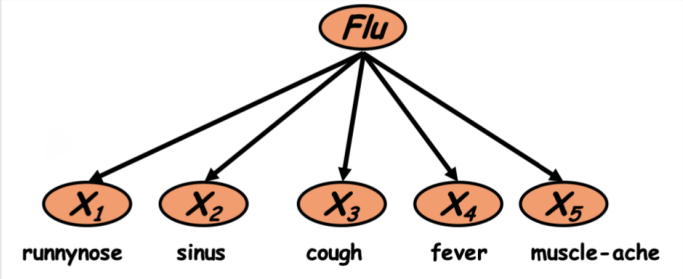
"A and B are independent"

if and only if

$$P(A|B) = P(A)$$

"A and B are independent" is often notated as

$$A \perp B$$



- Features (term presence) are *independent* of each other given the class:

$$P(X_1, ..., X_5|Y) = P(X_1|Y) \bullet P(X_2|Y) \bullet ... \bullet P(X_5|Y)$$

Top
Foot

Naive Bayes classifier: estimate the probability of each class given a text:

- Compute the posterior probability (Bayes rule) of each class c_i for text segment T

$$P(c_i|T) = \frac{P(T|c_i)P(c_i)}{P(T)}$$

- Assumption of independence between features ("naive" assumption)

$$P(T|c_i) = P(t_1, t_2, ..., t_j|c_i) \approx \prod_{j=1}^{n} P(t_j|c_i)$$

where T is described by a number of attributes or features $t_1, ..., t_j$

I.e. joint probability of the features given the class is approximated by the product of the probabilities of each feature given the class.

A Naive Bayes classifier (ctd)

- **Likelihood**: product of probabilities of each feature value of segment occurring with class c_i

$$\prod_{j=1}^{n} P(t_j|c_i)$$

- **Prior**: probability of segment having class c_i

$$P(c_i)$$

- **Evidence**: product of probabilities of features of segment – constant term for all classes, so can be disregarded:

$$\prod_{j=1}^{n} P(t_j)$$

Final decision:

$$\underset{c_i}{\text{argmax}} \prod_{j=1}^{n} P(t_j|c_i)P(c_i) \quad = \underset{c_i}{\text{argmax}}\, P(c_i) \prod_{j=1}^{n} P(t_j|c_i)$$

- Example
  - A Naive Bayes classifier - a worked out example (ctd)
    - **Features**: adjectives (bag-of-words)

Doc	Words	Class
1	Great movie, excellent plot, renowned actors	Positive
2	I had not seen a fantastic plot like this in good 5 years. amazing !!!	Positive
3	Lovely plot, amazing cast, somehow I am in love with the bad guy	Positive
4	Bad movie with great cast, but very poor plot and unimaginative ending	Negative
5	I hate this film, it has nothing original. Really bad	Negative
6	Great movie, but not...	Negative
7	Very bad movie, I have no words to express how I dislike it	Negative

  - Prior:
    - P(positive) = count(positive)/N = 3/7 = 0.43
    - P(negative) = count(negative)/N = 4/7 = 0.57

Top
Foot

- Likelihoods
  - $$\mathbf{P}(\mathbf{t_j}|\mathbf{c_i}) = \frac{\mathbf{count}(\mathbf{t_j}, \mathbf{c_i})}{\mathbf{count}(\mathbf{c_i})}$$
  - Count word t_j in class c_i / total words in that class

$$\begin{array}{ll|ll}
P(amazing|positive) & = 2/10 & P(amazing|negative) & = 0/8 \\
P(bad|positive) & = 1/10 & P(bad|negative) & = 3/8 \\
P(excellent|positive) & = 1/10 & P(excellent|negative) & = 0/8 \\
P(fantastic|positive) & = 1/10 & P(fantastic|negative) & = 0/8 \\
P(good|positive) & = 1/10 & P(good|negative) & = 0/8 \\
P(great|positive) & = 1/10 & P(great|negative) & = 2/8 \\
P(lovely|positive) & = 1/10 & P(lovely|negative) & = 0/8 \\
P(original|positive) & = 0/10 & P(original|negative) & = 1/8 \\
P(poor|positive) & = 0/10 & P(poor|negative) & = 1/8 \\
P(renowned|positive) & = 1/10 & P(renowned|negative) & = 0/8 \\
P(unimaginative|positive) & = 0/10 & P(unimaginative|negative) & = 1/8 \\
P(!!!|positive) & = 1/10 & P(!!!|negative) & = 0/8
\end{array}$$

Relative frequencies for prior (P(ci)) and likelihood($P(t_j | c_i$) makethe model in a Naive Bayes classifier.

- At decision (test) time, given a new segment to classify, this model is applied to find the most likely class for the segment $\mathbf{argmax}\,\mathbf{P}(\mathbf{c_i}) = \prod_{\mathbf{j=1}}^{\mathbf{nP}} (\mathbf{t_j}|\mathbf{c_i})$
- e.g.
  - Given a new segment to classify (**test time**):

Doc	Words	Class
8	This was a fantastic story, good, lovely	???

    - $P(positive) * P(fantastic|positive) * P(good|positive) * P(lovely|positive)$
      3/7 * 1/10 * 1/10 * 1/10 = 0.00043
    - $P(negative) * P(fantastic|negative) * P(good|negative) * P(lovely|negative)$
      4/7 * 0/8 * 0/8 * 0/8 = 0
    - sentiment = positive
  - Given a new segment to classify (**test time**):

Doc	Words	Class
9	Great plot, great cast, great everything	???

$P(positive) * P(great	positive) * P(great	positive) * P(great	positive)$

      3/7 * 1/10 * 1/10 * 1/10 = 0.00043
$P(negative) * P(great	negative) * P(great	negative) * P(great	negative)$

      4/7 * 2/8 * 2/8 * 2/8 = 0.00893
    - sentiment = negative
  - what if both the positive and negative value equals to zero

Top
Foot

- Add smoothing to feature counts (add 1 to every count).
  - Likelihoods $\mathbf{P(t_j|c_i)} = \frac{\mathbf{count(t_j,c_i)+1}}{\mathbf{count(c_i)+|V|}}$ where the $|V|$ is the number of distinct attributes in training(all classes)

    number fo classes is the number of the total documents
  - example
Doc	Words	Class
12	Boring movie, annoying plot, unimaginative ending	???
$P(positive) * P(boring	positive) * P(annoying	positive) * P(unimaginative	positive)$

      3/7 * ((0 + 1)/(10 + 12)) * ((0 + 1)/(10 + 12)) * ((0 + 1)/(10 + 12)) = 0.000040
$P(negative) * P(boring	negative) * P(annoying	negative) * P(unimaginative	negative)$

      4/7 * ((0 + 1)/(8 + 12)) * ((0 + 1)/(8 + 12)) * ((1 + 1)/(8 + 12)) = 0.000143
    - sentiment = negative
- Evaluation:
  - It's simple and will work well if data is not sparse
  - How can we improve?
    - Using all words (in Naive Bayes) works well in some tasks
    - Finding subsets of words may help in other tasks
    - Using only adjectives can be limiting. Verbs like hate, dislike; nouns like love; words for inversion like not; intensifiers like very
    - Pre-built polarity lexicons can be helpful
    - Negation is important
  - Can contrast direct opinions versus more complex comparative opinions:
    - Direct sentiment expressions on target objects
      - "the picture quality of this camera is great."
    - Comparisons expressing similarities or differences between objects,
      - e.g., "car x is cheaper than car y."
  - How do we quantify how well our Sentiment Analysis systems work?
    - Create experimental datasets (aka test corpora)
    - Compare (positive vs negative) system to human classifications
    - Compute metrics like

Top
Foot

$$\text{Accuracy} = \frac{\#\text{ correctly classified texts}}{\#\text{ texts}}$$

$$\text{Precision Pos} = \frac{\#\text{ texts correctly classified as positive}}{\#\text{ texts classified as positive}}$$

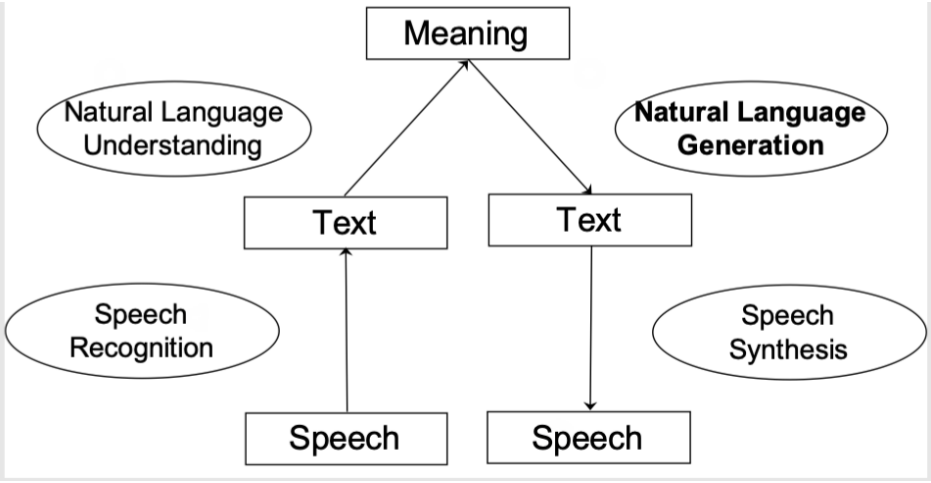
$$\text{Recall Pos} = \frac{\#\text{ texts correctly classified as positive}}{\#\text{ positive texts}}$$

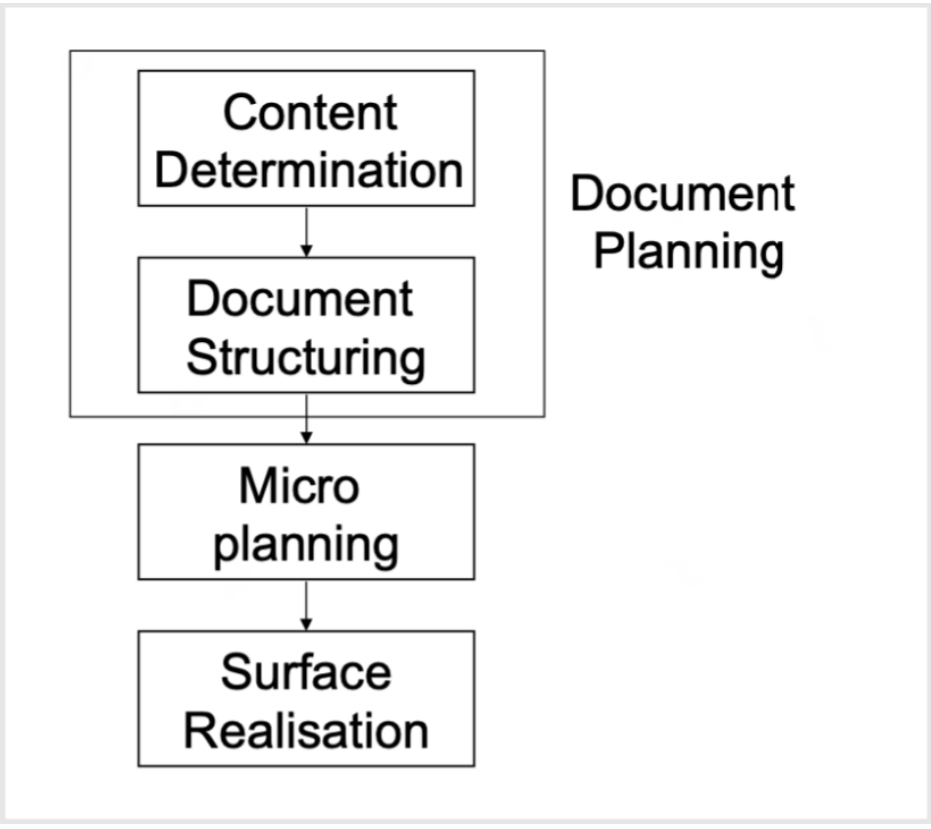
$$\text{F-measure Pos} = \frac{2 * \text{Precision Pos} * \text{Recall Pos}}{\text{Precision Pos} + \text{Recall Pos}}$$

3.Natural Language Generation

h3

Scanerio:





Goals:

- Decide on content: to determine what information to communicate

- Decide on rhetorical structure: to determine how to structure the information to make a coherent text

How to Choose Content

h4

- Theoretical approach: deep reasoning based on deep knowledge of user, task, context, etc
- Pragmatic approach: write schemas which try to imitate human-written texts in a corpus
- Statistical approach: use learning techniques to learn content rules from corpus

Theoretical Approach

h4

- Deduce what the user needs to know, and communicate this
- in-depth knowledge
  - User(knowledge,task,etc)
  - Context, domain,world
- AI
  - applies logical rules to the knowledge base to deduce new information
- Evaluation
  - Lack knowledge about user
  - Lack knowledge of context
  - Hard to do in practice because we don't have good models of the effects of choices
  - Very hard to maintain knowledge base like new usersm new regulations, etc.

Statistical Approach

h4

- Statistical/learning techniques (including deep learning)
  - Parse corpus, align with source data, use machine learning algorithms to learn content selection rules/schemas/cases
- Worth considering if large corpora available

Pragmatic Approach: Schema

h4

- Analyse corpus texts (after aligning them to data), and manually infer content and structure rules.
- Typically based on imitating patterns seen in human-written texts
  - Revised based on user feedback
- Specify structure as well as content
- Evaluation:
  - Sometimes corpus texts may not be very good from a microplanning perspective

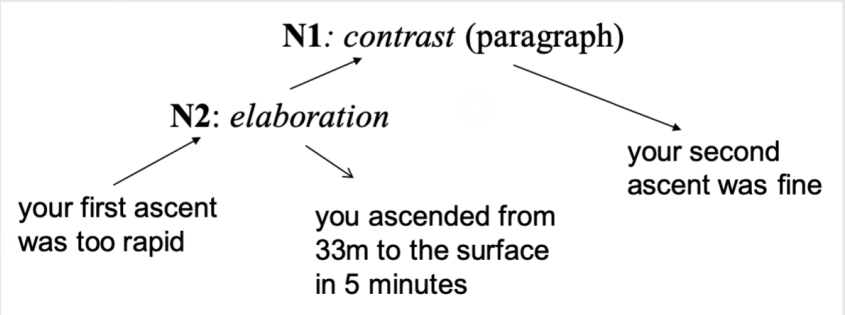
Text structure

h4

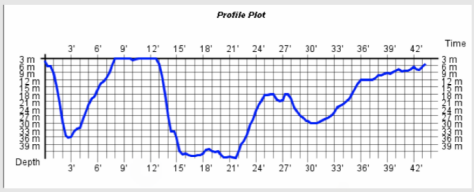
- Rhetorical Relations: describe how the parts of a text are linked to each other.
- Example:
  -

Top
Foot

- Your first ascent was too rapid; you ascended from 33m to the surface in 5 minutes. However, your second ascent was fine.



diveNo	segNo	iTime	iValue	fTime	fValue
1460	1	0	1.3	60	6.3
1460	2	60	6.3	140	32.2
1460	3	140	32.2	480	0
1460	4	480	0	760	0
1460	5	760	0	920	38.9
1460	6	920	38.9	1300	41.6
1460	7	1300	41.6	1500	15.5
1460	8	1500	15.5	1860	27.2
1460	9	1860	27.2	2160	9.2
1460	10	2160	9.2	2600	2.7



Input: 1460 3 140 32.2 480 0
Output: (representation of)
Your first ascent was a bit rapid; you ascended from 33m to the surface in 5 minutes, it would have been better if you had taken more time to make this ascent.

Input: 1460 10 2160 9.2 2600 2.7
Output: (representation of)
Your second ascent was fine.

Lexical Choice

h4

- the task of choosing the right words or lemmas to express the contents of the message
- Issues:
  - Frequency (affects readability)
  - Formality
  - Focus, expextations
  - Technical terms

Top
Foot

- Convention

> Statistics-Based Lexical Choice for NLG from Quantitative Information h4

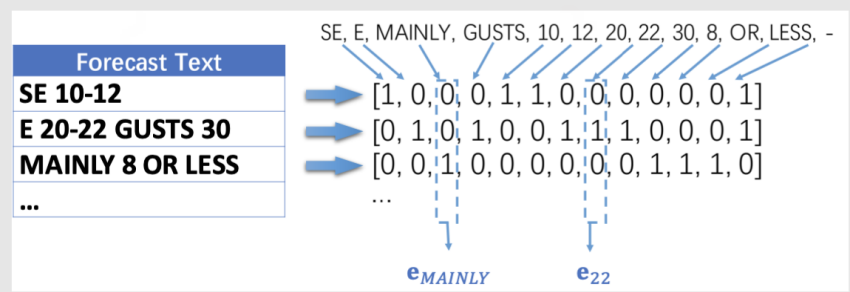
- Goal
  - Systems need to “know” what expressions are most suitable for expressing a given piece of information.
  - To develop a statistical algorithm for lexical choice for quantitative information, which can
    - Detect the relationship between data dimensions (aka. attributes) and words
    - Does not rely on hand-crafted rules;
    - Predict both when and which words should be used
    - One word can refer to multiple dimensions
-
  - Each data record consists of attribute-value pairs.
  - E.g., dir=2,ws=9,gusts=11, where the attributes are “dir”, “ws”, and “gusts”.

Forecasted Wind Direction (azimuth)	numerical Wind Speed (knots)	data Gust (knots)
2	9	11
92	20	30
130	4	5

- Methodology(Vector)
  - We represent each attribute (e.g. wind speed) as a combination of some weighted key-points.
    - Taking the min and max values of the attribute (from training data)
    - Key-points are evenly spaced between the min and max values
  - The number of the key-points for an attribute are fixed
  - An example of deriving the key point weights for the attribute value ws=9 (i.e., wind speed dimension).
    - 44
  - Similarly, a data record (i.e., a set of attribute-value pairs) can be represented by multiple groups of key-points, e.g.:
    - ws = 9 → [0.1, 0.9, 0, 0, 0]
    - dir = 2 → [0.97, 0.03, 0, 0, 0]
  - Thus, to represent a set of attribute-value pairs, we concatenate the individual weight vectors, e.g.:
    - {ws = 9, dir = 2} → [0.1, 0.9, 0, 0, 0, 0.97, 0.03, 0, 0, 0]
  - The entire data-text corpus can then be represented with a vector matrix (K), whose row corresponds to the weight vector of a data record.
    -

$$\begin{matrix} & & & & & & \overbrace{\quad\quad\quad\quad\quad}^{\text{Wind speed}} & & & & & \overbrace{\quad\quad\quad\quad\quad}^{\text{Wind direction}} & \\ \{ws=9, dir=2, \dots\} & & & & & & & & & & & & \\ \{ws=20, dir=130, \dots\} & \rightarrow \boldsymbol{K} = & \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 & 0.97 & 0.03 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.66 & 0.44 & 0 & 0 & \dots \\ 0.8 & 0.2 & 0 & 0 & 0 & 1 & 0.86 & 0.14 & 0 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} & \\ \{ws=2, dir=90, \dots\} & & & & & & & & & & & & \\ \dots & & & & & & & & & & & & \end{matrix}$$

Top
Foot

- We use a column vector (namely ei)to represent the text of a datarecord Each element of ei indicates whether a word appears in the data record, e.g.:
  - 

- We represent words in the corpus using the same weight vectors whose values are unknown.
- To estimate vi for word i given a data-to-text corpus as input
- v_i and v_d should be close to each other in the vector space if word i appears in data record d
  - $$\frac{v_{d1} \cdot v_i}{\|v_{d1}\|\|v_i\|} = appear(i, d1)$$
    $$\frac{v_{d2} \cdot v_i}{\|v_{d2}\|\|v_i\|} = appear(i, d2)$$
    $$\ldots$$
  - appear $(i, d1) = 1$ if word i appears in data record d and 0 otherwise.
- Our task is to find the weight vector vi for each word i, such that the similarity of v_i and v_d is close to appear (i, d) as much as possible for each data record (d).
  - $$v_i = \min_{v_i} \sqrt{\sum_d (sim(v_i, v_d) - appear(i, d))^2}$$

Microplanning

h4

- Decide how to best express a message in language
- Imitating corpus works to some degree, but not perfectly
- Key is better understanding of how linguistic choices affected readers
  - Our SumTime weather-forecast generator microplans better than human forecasters

Relisation

h4

- Creating linear text from (typically) structured input; ensuring syntactic correctness
- Take care of details of language
- Problem
  - There are lots of finicky details of language which most people developing NLG systems don't want to worry about
  - Solution:
    - Automate this using a realiser

Top
Foot

Morphology

h4

- In linguistics, morphology is the study of words, how they are formed, and their relationship to other words in the same language.
- Variations of a root form of a word
- Inflectional morphology
- Derivational morphology - change meaning

4.Information Extraction(IE)

h3

Definition

From each text in a set of unstructured natural language texts identify information about predefined classes of entities, relationships or events and record this information in a structured form by either:

- Annotating the source text, e.g. using XML tags;
- Filling in a data structure separate from the text, e.g a template or database record or “stand-off annotation”
- The activity of populating a structured information repository (database) from an unstructured, or free text, information source
- The activity of creating a semantically annotated text collection (cf.“The Semantic Web”)

Purpose:

- searching or analysis using conventional database queries
- data-mining;
- generating a summary (perhaps in another language);

Task

- Given: a document collection and a predefined set of entities, relations and/or events
- Return: a structured representation of all mentions of the specified entities, relations and/or events

Evaluation:

- Strengths
  - Extracts facts from texts, not just texts from text collections
  - Can feed other powerful applications(databases, semantic indexing engines, data mining tools)
- Weakness:
  - Systems tend to be genre/domain specific and porting to new genres and domains can be time-consuming/requires expertise
  - Limited accuracy
  - Computationally demanding, so performance issues on very large collections

Entity Extraction

h4

Top
Foot

- For each textual mention of an entity of one of a fixed set of types identify its extent and its type
- Coreference
  - Multiple references to the same entity in a text are rarely made using the same string:
    - Pronouns – Tony Blair . . . he
    - Names/definite descriptions - Tony Blair
    - Abbreviated forms
    - Orthographic variants
  - Different textual expressions that refer to the same real world entity are said to corefer.
  - Clearly IE systems are more useful if they can recognise which text mentions are coreferential.
  - **Coreference Task**: link togther all textual references to the same real world entity, regardless of whether the surface form is a name or not

Relation Extraction

h4

- Task:
  - identify all assertions of relations, usually binary, between entities identified in entity extraction
- May be divided into two subtasks:
  - Relation detection: find pairs of entities between which a relation holds
  - Relation classification: for pairs of entities between which a relation holds, determine what the relation is
- Challenging
  - Discovering relations frequently depends upon being able to follow coreference links.
  - The same relation may be expressed in many different ways:

Event Detection

h4

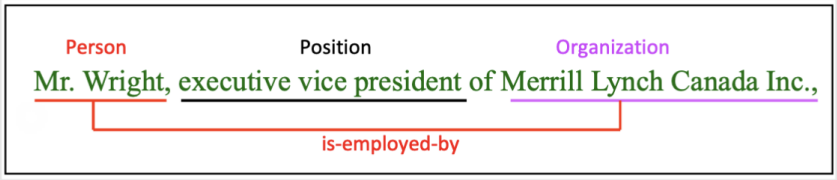
- Task
  - identify all reports of event instances, typically of a small set of classes
- May be divided into two subtasks
  - Event detection: find mentions of events in text
  - Event classification: assign detected events to one of a set of classes
- Events may be simply viewed as relations. However they are typically complex relations that
  - Are temporally situated and often of relatively short duration
  - Involve multiple role players
  - Are often expressed across multiple sentences

Approach

h4

- Knowledge Engineering Approaches
- Supervised Learning Approaches
- Bootstrapping Approaches
- Distant Supervision Approaches

Knowledge Engineering Approaches



- Such systems use manually authored rules and can be divided into
  - "deep" – linguistically inspired "language understanding" systems
  - "shallow" – systems engineered to the IE task, typically using pattern-action rules

    Pattern: ''Mr. $Uppercase-initial-word''
    Action: add-entity(person("Mr. $Uppercase-initial-word))

    Pattern: \$Person, $Position of $Organization"
    Action: add-relation(is-employed-by($Person,$Organization))

Supervised Learning Approach

- Systems are given texts with manually annotated entities + relations
- For each entity/relation create a training instance
  - k words either side of an entity mention
  - k words to the left of entity 1 and to the right of entity 2 plus the words in between
- Training instances represented in terms of features
  - words, parts of speech, orthographic characteristics, syntactic info
- Systems may learn
  - patterns that match extraction targets
  - Classifiers that classify tokens as beginning / inside / outside a tage type
- Learning techniques include: covering algorithms, HMMs, SVMs

Bootstrapping Approaches

- A technique for relation extraction that requires only minimal supervision
- Systems are given
  - seed tuples
  - seed patterns
- System searches in large corpus for
  - occurences of seed tuples and then extracts a pattern that matches the context of the seed tuple
  - matches of seed patterns from which it harvests new tuples
- New tuples are assumed to stand in the required relation and are added to the tuple store

Distant Supervision Approaches

- Sometimes also called "weakly labelled" approaches
- Assumes a (semi-)structured data source, such as
  - which contains tuples of entities standing in the relation of interest and, ideally, a pointer to a source text
- Tuples from data source are used to label
  - the text with which they are associated, if available
  - documents from the web, if not

Top
Foot

- Labelled data is used to train a standard supervised named entity or relation extraction system

Evaluation

h4

- **keys**
  - Correct answers, called keys, are produced manually for each extraction task (filled templates or SGML annotated texts)
- **responses**
  - Scoring of system results, called responses, against keys is done automatically.
- **interannotator agreement**
  - At least some portion of the answer keys are multiply produced by different humans so that interannotator agreement figures can be computed.
- Principal metrics – borrowed from information retrieval
  - Precision (how much of what system returns is correct)
  - Recall(how much of what is correcrt system returns)
  - F-measure(a weighted combination of precision and recall)
- **Shared Task challenges**
  - Shared Task challenges are community wide exercises in which groups of researchers engage in a friendly competition to build systems to address a common task.

Named Entity Recognition

h4

- **Tasks**
  - for each textual mention of an entity of one of a fixed set of types identify its extent and its type.
- **Recap**
  - Types of entities which have been addressed by IE systems
    - Named individuals
      - Organisations, persons, locations, books, films, ships, restaurants . . .
    - Named Kinds
      - Proteins, chemical compounds/drugs, diseases, aircraft components . . .
    - Times
      - temporal expressions – dates, times of day
    - Measures
      - temporal expressions – dates, times of day
  - Multiple references to the same entity in a text are rarely made using the same string:
    - Pronouns – They . . . he
    - Names/defiinite descriptions - Tony Blair . . . the Prime Minister
    - Abbreviated forms - Theresa May … May; United Nations … UN
    - Orthographic variants
  - Different textual expressions that refer to the same real world entity are said to corefer
  - **Coreference Task**: link togther all textual references to the same real world entity, regardless of whether the surface form is a name or not

Knowledge Engineering Approaches to NER

- Such systems typically use

Top
Foot

- named entity lexicons and
- Manually authored pattern / action rules or regular expression

- System has three main stages:
  - Lexical processing
  - NE parsing
  - Discourse interpretation

- Lexical processing
  - Many rule-based NER systems made extensive use of specialised lexicons of proper names, such as gazetteers – lists of place names
  - Why not use even larger gazetteers?
    - Many NEs occur in multiple categories - the larger the lexicons the greater ambiguity
    - the listing of names is never complete, so need some mechanism to type unseen NEs in any case
  - Principles
    - Tokenisation, sentence splitting, morphological analysis
    - Part-of-speech tagging
      - tags known proper name words andunknown uppercase-initial words as proper names (NNP, NNPS) NNP = Proper nouns
    - Name List / Gazetter Lookup and Tagging (organisations, locations, persons, company designators. person titles)
    - Trigger Word Tagging
      - certain words in multi-word names function as trigger words, permitting classification of the name
      - system has trigger words for various orgs,gov't institutions,locations

- NE Parsing
  - Using the presetrules to identify the unclassified proper name
  - - After lexical processing the next step in the Wakao et al. system is NE parsing.
    - The system has 177 hand-produced rules for proper names: 94 for organisation; 54 for person; 11 for location; 18 for time expressions.
    - A fragment of the proper name grammar:

      ```
      NP--> ORGAN_NP
      ORGAN_NP --> LIST_LOC_NP NAMES_NP CDG_NP
      ORGAN_NP --> LIST_ORGAN_NP NAMES_NP CDG_NP
      ORGAN_NP --> NAMES_NP '&' NAMES_NP
      NAMES_NP --> NNP NAMES_NP
      NAMES_NP --> NNP
      ```

    - The rule ORGAN NP --> NAMES_NP '&'' NAMES_NP means:
      If an unclassified proper name (NAMES_NP) is followed by '&' and another unclassified proper name, then it is an organisation name.
      E.g. Marks & Spencer and American Telephone & Telegraph

- Discourcse Interpretation - Coreference Resolution
  - When the name class of an antecedent (anaphor) is known thenestablishing coreference allows the name class of the anaphor (antecedent) to be established.
  - An unclassified PN may be co-referential with a variant form of a classified PN. (PN = Pronoun)
    - Ford – Ford Motor Co.
  - An unclassified PN may be co-referential with a definite NP which permits the PN's class to be inferred
    - E.g. Kellogg … the breakfast cereal manufacturer

Top
Foot

    - Semantic type information
        - noun-noun qualification
            - When an unclassified PN qualifies an organisation-related object then the PN is classified as an organisation; e.g. Erickson stocks
        - Possessives
            - when an unclassified PN stands in a possessive relationto an organisation post, then the PN is classified as an organisation; e.g. vice president of ABC, ABC's vice president
        - Apposition
            - when an unclassified PN is apposed with a known orgnisation post, the former name is classified as a person name; e.g. Miodrag Jones, president of XYZ
- Evaluation
    - Strengths
        - High performance - only several points behind human annotators
        - Transparent - easy to understand what system is doing/why
    - Weakness
        - Porting to another domain requires substantial rule re-engineering
        - Acquisition of domain-specific lexicons
        - Rule writing requires high levels of expertise

Supervised learning approaches to NER

- Task
    - Supervised learning approaches aim to address the portability problems inherent in knowledge engineering NER
        - Instead of manually authoring rules, systems learn from annotated examples
        - Moving to new domain requires only annotated data in the domain – can be supplied by domain expert without need for expert computational linguist
- Sequence Labelling
    - Systems may learn
        - patterns that match extraction targets
        - classifiers that label tokens as beginning/inside/outside a tage type
    - In sequence labelling for NER, each token is given one of three label types:
        - 
        - Scheme is called `BIO` or sometimes `IOB`
    - Each training instance(token) is typically represented as a set of **features**
    - **Features** can be not only the characteristics of the token itxself but of neighbouring tokens as well
        - the classifier extracts features from
            - input string
            - left predictions
- Carreras et al.(2003)
    - NE detection: in a first pass over the text BIO tags are assigned without regard to type.
    - NE classification: in a second pass the NE's detected in the first pass are assigned a class (organisation, person, location, etc.)
    - Adaboost classifier
    - Type pattern of consecutive words in context
        - functional (f)
        - capitalized (C)

Top
Foot

        - lowercased (l)
        - punctuation mark (.)
        - quote ()
        - other(x)
- **Entity Linking**
    - KBP: Knowlege base population
        - Facts are gathered from open access web sources and used to build a structuted information repository.
    - Entity Linking Task: Given a text with a recognised NE mention in that text and a knowledge base (KB), such as Wikipedia, link the NEs to the matching entry in the KB if there is one, else create an entry.
        - Very Difficult Task
    - Approach:
        - Given a text T containing an NE mention m and using Wikipedia as a KB
            - index all pages in the KB using an information retrieval system
            - Build a query from T (e.g. use the sentence/paragraph/whole text) containing m and search the KB
            - From the ranked list of KB pages returned by step 2 pick the high ranked page whose name matches m and return it
        - Doesn't work very well

- Conclusion:
    - Named Entity Recognition (NER) is a core IE technology that is now relatively mature and at "usable" performance levels.
    - NER aims to detect and classify all mentions of named entities of a given set of entity types within a given text.
    - Techniques used have included:
        - Knowledge engineering approaches
        - Supervised Learning Approach
    - Challenge:
        - Reducing the amount of training data needed via, e.g. bootstrapping techniques.
        - Exploiting existing structured data sources to generate "weakly labelled" training data (aka distant supervision)
        - Expanding the classes of entities addressed
        - Developing NERs for languages other than English.

Relation Extraction

- Task:
    - given a text T and a set of relations R, identify all assertions of relations from R in T, holding between entities identified in entity extraction.
- Relations in R are usually binary
- Reations in are assumed to be a subset of those identifed in the entity extraction process
- Challenge:
    - The same relation may be expressed in many different ways:
    - The information required may be spread across multiple sentences and discovering relations may depend upon following coreference links.

- The information to be extracted may be implied by the text, rather than explicitly asserted, and extracting it may require inference

Knowledge Rngineering Approach

Use the authored rules and can be divided into

- Shallow: using the pattern-action rules
- Deep: linguinstically inspired “language understanding” systems
  - typically parse input using broad coverage NL parser to identify key grammatical relations, like subject and object
  - use transduction rules to extract relations of interest from parser output
  - Extraction rules over parser output allow a wider set of expressions to be captured than with regex’s over words and NE tags alone

Strengths:

- high precision
- System behaviour is human-comprehensible

Weakness:

- The writing of rules has no end
- New rules needed for every new domain(pattern action rules for shallow approaches; transduction rules for deep approaches)

Suppervised Learning Approaches

What to be learned?

- Rules that
  - Match to all and only relation bearing sentences
  - Capture sunstrings within the matched text that correspond to relation arguments
- Binary classifier
  - containing instances of the entity types between which the relation holds
  - As with NER can be divided into detection and classification stages:

Process:

- Assume entities to be related already tagged
- Use any algorithm for learning binary classifiers to learn to distinguishinstances (typically sentences) where
  - entities co-occur and relation holds (positive instances)
  - entities cooccur anreation os (positive instances)
- Features used fall into 3 broad classes:
  - Features of the named entities
  - Features from the words in the text, usually words from 3 locations
  - Features about the entity pair within the sentence

Strengths:

- No need to write extensive/complex rule sets for each domain
- Same system straightforwardly adapts to any new domain, provided training data is supplied.

Weakness:

Top
Foot

- Quality of relation extraction dependent on quality and quantity of training data, which can be difficult and time consuming to generate
- Developing feature extractors can be difficult and they may be noisy (e.g. parsers) reducing overall performance

Bootstrapping Approaches

Motivation:

- reduce number of manually labelled examples needed to build a system

Key Idea:

- set of trusted tuples T (e.g. pairs of entities known to stand in the relation of interest)
- set of trusted patterns P (i.e. patterns known to extract pairs of entities in the given relation with high accuracy)

  The if
  - then find tuples from T in sentences S in D, extract patterns from context of sentences in S, add patterns to P and then use P to find new tuples in D and add to T; repeat until convergence.
  - then match patterns from P in sentences S in D, extract tuples from pattern matches in sentences in S, add tuples to T and then use tuples in T to find new patterns in D and add to P; repeat until convergence.

DIPRE(Dual Iterative Pattern Relation Expansion)

- Aim
  - to extract useful relational tuples from the Web, of the form
- Method
  - Exploit “duality of patterns and relations”
    - Good tuples help find good patterns
    - Good patterns help find good tuples
  - Starting with user-supplied tuples, iteratively
    - Use these tuples to find patterns
    - Use the patterns to find more tuples
- Patterns are defined as 5-tuples: (order, urlprefix, prefix, middle,suffix)
- Occurrences are defined as 7-tuples: (author,title, order, url, prefix, middle,suffix)
- An algorithm for generating a pattern given a set of occurrences is described
  - Algorithm inisists order and middle of all occurrences is the same – they form part of the generated pattern
  - Additionally pattern contains
    - longest matching prefix of the url of all the occurrences
    - longest matching suffix of the prefix of all the occurrences
    - longest matching prefix of the suffix of all the occurrences
    - See Brin (1999) for details
- Patterns are assessed for specificity and rejected if their specificity is too low, i.e. if they are too general.
- Conclusion
  - Strengths:
    - Need for manually labelled training data is eliminated

Top
Foot

- Weaknesses:
  - Can suffer from semantic drift – when an erroneous pattern introduces erroneous tuples, which in turn lead to erroneous patterns
  - Works well when significant redudancy in assertion of specific tuples an in use of specific patterns to express a relation
  - Issues when multiple relations hold between the same pair of entities

Distant Supervision Approaches

Aim:

- reduce/eliminate the need for manually labelled training data

Key Idea:

- Suppose we have a large document collection D plus a structured data source (e.g. a database) R that contains
  - many instances of a relation of interest in, e.g., a relational table
  - optionally, for each relation instance a link to a document in D providing evidence for the relation

Method:

- search for sentences in D containing the entity pairs that occur in relation instances (tuples) in R
- label these sentences as positive occurrences of the relation instance
- use the labelled sentences as training data to train a standard supervised relation extractor

Freebase - a free on-line database of structured semantic data

Distant Supervision Assumption

if two entities participate in a relation, any sentence that contains those two entities might express that relation.

- combine features from multiple mentions to get a richer feature vector
- use multiclass logistic regression as a learning framework
- At test time features are combined from all occurrences of a given entity pair in the test data and the most likely relation is assigned

Negative instances

- to get these, randomly select entity pairs that do not appear in any Freebase relation and extract features for them
- Freease reation anextract eatures or tem these rare occurrences should be low

Conclusion:

- Strengths:
  - Need for manually labelled training data is eliminated
  - Can very rapidly get extractors for a wide range of relations
- Weakness:
  - Precision still lags behind best knowledge-engineered/directly supervised learning approaches
  - Only works if a good supply of structured data is available for the relation(s) of interest

Top
Foot

Past Paper

2013-2014

Section A

a) In the context of Information Retrieval, explain the difference between algorithms that perform boolean search and algorithms that perform a ranked search. What type of algorithm would be better for a regular user (such as an undergraduate student in the Humanities area) who is using a search query with multiple terms, which he/she expects to appear in many documents? Explain the reasons behind your choice of algorithm type.

Answer:

For the boolean search model, the model just needs to decide the whether the document is relevant or not. And the presence of the terms is very essential and suffient for the search. For the operator, the boolean search model exploits the boolean operators like AND and OR. The boolean query provides the logical result for deciding whether the document should be returned.

For the ranked search method like the vector space model, it relys on the frequency of terms and maybe some weight of terms could affect the result of the search. The documents are point in high-dimension vector space and its value is the frequency of the terms. Also, the query should be represented as the vectors. This method record the vectors for each documents and queries, finally calculate the similarity metrics which can be interpreted as the normalised correlation coefficient.

For the choice of the searching algorithm, the ranked algorithm should be chosen. Because the boolean algorithm is difficult and unnatural for the newbies. And the users don't want the plenty of ranked results list. For the ranked algorithm, it's very easy to generate the result lists and pick up the relevant documents.

b) Compression techniques are important due to the growth in volume of the data that must be stored and transmitted.

(i) Explain the difference between lossy and lossless forms of compression. Dis□cuss the suitability of these alternative forms of compression for different media types (e.g. for text vs. image data).

Answer:

Data Compression is a technique in which the size of data is reduced without loss of information. **Lossy compression** and **Lossless compression** are the categories of data compression method.

The main difference between the two compression techniques (lossy compression and Lossless compression) is that, The lossy compression technique does not restored the data in its original form, after decompression on the other hand lossless compression restores and rebuilt the data in its original form, after decompression.

S.NO	LOSSY COMPRESSION	LOSSLESS COMPRESSION
1.	Lossy compression is the method which eliminate the data which is not noticeable.	While Lossless Compression does not eliminate the data which is not noticeable.
2.	In Lossy compression, A file does not restore or rebuilt in its original form.	While in Lossless Compression, A file can be restored in its original form.
3.	In Lossy compression, Data's quality is compromised.	But Lossless Compression does not compromise the data's quality.
4.	Lossy compression reduces the size of data.	But Lossless Compression does not reduce the size of data.
5.	Algorithms used in **Lossy** compression are: Transform coding, Discrete Cosine Transform, Discrete Wavelet Transform, fractal compression etc.	Algorithms used in **Lossless** compression are: Run Length Encoding, Lempel-Ziv-Welch, Huffman Coding, Arithmetic encoding etc.
6.	Lossy compression is used in Images, audio, video.	Lossless Compression is used in Text, images, sound.
7.	Lossy compression has more data-holding capacity.	Lossless Compression has less data-holding capacity than Lossy compression technique.

(ii)Explain the difference between static, semi-static and adaptive techniques for text compression, noting their key advantages and disadvantages.

Answer:

A static model is a fixed model that is known by both the encoder and thedecoder and does not depend on the specific data that is beingcompressed.

A semiadaptive or semistatic model is a fixed model that is constructed from the data to be compressed.

An adaptive model changes during the compression. At a given point incompression, the model is a function of the previously compressed part ofthe data. Since that part of the data is available to the decoder at thecorresponding point in decompression, there is no need to store the model.

c)The two main model components in Statistical Machine Translation are the Translation Model and the Language Model. Explain the role of each of these components.Describe the type of data that is necessary to build each of them. Mention one way in which these components can be combined to build a translation system.

**d)Assume we have a small set of seed words with positive and negative opinions, e.g.: positive = {good, fast, cheap} and negative = {slow, boring, fragile}. Explain the two most common (semi-)automated approaches to expand these sets with more opinion words or phrases to create lexica for Sentiment Analysis, providing examples whenever possible. Give one advantage and one disadvantage of each approach. **

Answer:

This model is created from the seed words which comes from the supervised seed positive and negative words. It can be divided into two parts ways, Dictionary-based and Corpus-based approach. For the

Top
Foot

Dictionary-based, the users should find the synonyms/antonyms of seed emotion words in dictionaries like WordNet. Otherwise, users can build the patterns from the seed words/phrases to search on large corpora. For the Corpus-based approach, the examples are annotated with sentiment are used withe machine learning algorithms to learn a classifier for each sentence and document. The users can rely on the manully way (gold-standards) and crow-annotated resources like Amazon Product Resource. This approach can be divided into the two steps: subjectivity classifier : first run binary classifier to identify and then eliminate objective segments. Subjectivity Classifier with remaining segments: learn how to combine and weight different attributes to make predictions.

For the advantages for the Dictionary-based approach, it doesn't need the labelled data and the procedure of learning is not required. For disadvantages, it requires powerful linguistic resources which is not always available, so users cannot guarantee the efficiency of this approach.

For the advantages for the Corpus-based approach, it really easy to implement anf usually works well. As for the disvantage, the assumption of this approach is hard to estimate.

Section B

2.In the context of Information Retrieval, given the following documents:

Document 1: Sea shell, buy my sea shell! Document 2: You may buy lovely SEA SHELL at the sea produce market. Document 3: Product marketing in the Shelly sea is an expensive market. and the query: Query 1: sea shell produce market

a)Apply the following term manipulations on document terms: stoplist removal, capi□talisation and stemming, showing the transformed documents. Explain each of these manipulations. Provide the stoplist used, making sure it includes punctuation, but no content words.

Answer:

For the stop removal list, it should exclude the "non-content" words, so the stop list should be "my, may ,at, you ,the, in, is, an, , , ! , .".

For the capitalisation, turn all the words to lower case.

For the stemming, turn the `marketing` to `market` . `product` to `produce`

So the transformed document should be:

Document 1: sea shell buy sea shell

Document 2: buy lovely sea shell sea produce market

Document 3: produce market shelly sea expensive market

b)Show how Document 1, Document 2 and Document 3 would be represented using an inverted index which includes term frequency information.

Answer:

sea: (1,1) (1,4) (2,3) (2,5) (3,4)

shell: (1,2) (1,5) (2,4)

buy: (1,3) (2,1)

lovely:(2,2)

Top
Foot

produce: (2,6) (3,1)

market: (2,7) (3,2) (3,6)

shelly:(3,3)

expensive: (3,5)

c)Using term frequency (TF) to weight terms, represent the documents and query as vectors. Produce rankings of Document 1, Document 2 and Document 3 according to their relevance to Query 1 using two metrics: Cosine Similarity and Euclidean Distance. Show which document is ranked first according to each of these metrics.

Answer:

TF:

Query: sea(1) shell(1) produce(1) market(1)

Document1: sea(2) shell(2) produce(0) market(0) buy(1)

Document2: sea(2) shell(1) produce(1) market(1) buy(1) lovely(1)

Document3: sea(1) shell(0) produce(1) market(2) expensive(1)

Evaluation:

Similarity cosine:

Document1: (2×1+2×1+0×1+0×1+0×1) / (2× √(4+4+0+0+1)) = 4/6

Document2: (2×1+1×1+1×1+1×1+0×1+0×1) / (2×√(4+1+1+1+1+1)) = 5/6

Document3: (1×1+0×1+1×1+2×1+0×1) / (2×√(1+0+1+4+1) = 4/5.29

Euclidean Distance:

Document1: √(1+1+1+1+1) = √5

Document2: √(1+0 +0 + 0 + 1 + 1) = √3

Document3: √(0+1+0+1+1) = √3

For the best similarity, we should choose the lowest value document.

d)Explain the intuition behind using TF.IDF (term frequency inverse document fre□quency) to weight terms in documents. Include the formula (or formulae) for com□puting TF.IDF values as part of your answer. For the ranking in the previous question using cosine similarity, discuss whether and how using TF.IDF to weight terms instead of TF only would change the results.

Answer:

For using the Terms frequency, the result of the model may be affected by the freqency of the terms in documents, so we should consider the less common terms could be more useful to find the relevant documents, so we use the inverse document frequency to avoid this situation happening.

we need the document frequency (df) which include the key terms from the query.

sea : idf:0 log1 = 0

shell: idf: log(3/2)=0.17609125905

buy: idf: log(3/2)=0.17609125905

lovely: idf: log(3/1)=0.477

produce: idf: log(3/2)=0.17609125905

market: idf: log(3/2)=0.17609125905

shelly: idf: log(3/1) = 0.477

expensive: idf: log(3/1)=0.477

Evaluation:

Terms	Query	Document1	Document2	Document3
Sea	0×1/4	0×2/5	0×2/6	0×1/5
Shell	0.18×1/4	0.18×2/5	0.18×1/6	0.18×0
Buy	0	0.18×1/5	0.18×1/6	0.18×0
Lovely	0	0.477×0	0.477×1/6	0.477×0
Produce	0.176×1/4	0.176×0	0.176×1/6	0.176×1/5
Market	0.176×1/4	0	0.176×1/6	0.176×1/5
Shelly	0	0	0	0.477×1/5
Expensive	0	0	0	0.477×1/5
	0.20925	0.108	0.1975	0.0892

Document1: (0+(0.045×0.072)) / 0.20925×0.108

Document2: (0+(0.03×0.045)+(0.044×0.0293)×2) / 0.20925×0.1975

Document3: (0+(0.044×0.0352)) / 0.20925×0.0892

e)Explain the metrics Precision, Recall and F-measure in the context of evaluation in Information Retrieval against a gold-standard set, assuming a boolean retrieval model. Discuss why it is not feasible to compute recall in the context of searches performed on very large collections of documents, such as the Web.

Answer:

Recall: the proportion of the relevant documents

Precision: the proportion of retrieved documents that are relevant

Precision and Recall address the relation between the retrieved and relevant sets of documents.

F-measure: combines precision and recall into a single figure,gives equal weight to both. It is the F is a harmonic mean which penalises low performance in one value more than arithmethic mean.

Because the there are tremendous web pages which is included in the Internet, the mount of the retrieved pages is pretty small because of the assumption and limitation of algorithms.

4.

Top
Foot

a) Differentiate subjectivity from sentiment. How are the tasks of Subjectivity Classification and Sentiment Analysis related?

Answer:

As for the rule-based subjectivity classifier, the task is to search for the emotion words lexicon and determine the sentence/document is objective or subjective.

As for the rule-based sentiment classifier, the task is to determine the document/sentence shows the positive sentiment or negative sentiment by counting the value of lexicons and build the judgement model.

b)Explain the steps involved in the lexicon-based approach to Sentiment Analysis of features in a sentence (e.g. features of a product, such as the battery of a mobile phone). Discuss the limitations of this approach.

Answer:

As for the binary approach, the input is the sentences s and product features f, the output is the attitude of this feature is positive,negative or netural.

Step1: the model should pick up all the sentences which contains the fratures and lexicons about the attitude. Step2: the model should select all the emotion words in sentence Step3: assign the values of emotion words, 1=positive, 0=netural, -1=negative Step4: sum up the orientation and assign the orientation to (f,s)

As for the shortcoming for intensifiers, the gradable approach assign the different levels to the emotional content. The process is similar to the binary approach, the final decision is based on the all emotion words.

The disadvantage is requiring a lexicon of emotion words which should be fairly comprehensive,covering new words.abbreviations,misspelled words.

c)Explain the graded lexicon-based approach for Sentiment Analysis.Given the following sentences and opinion lexicon (adjectives only), apply the weighted lexical-based approach to classify EACH sentence as positive, negative or objective. Show the final emotion score for each sentence. In addition to use of the lexicon, make sure you consider any general rules that have an impact in the final decision. Explain these rules when they are applied.

Lexicon: boring -3 brilliant 2 good 3 horrible -5 happy 5

Graded lexicon-based Approach:

(S1)He is brilliant but boring.**

Diminisher rule: the weight should be substracted from the positive terms.

emotion(brilliant) = 2 emotion(boring) = -3 , so the decision value is 2-3 = -1

(S2) I am not good today.

Negation rule: when the neighbourhood area occurs the negation words, the value should be decreased by 1 and inverted.

emotion(good) = 3, the decision value is -(3-1) = -2

Top
Foot

(S3) I am feeling HORRIBLE today, despite being happy with my achievement.

Capitalization rule : the value of emtions words should be increased by 1 for positive words,-1 for negative words. Diminisher rule: the weight should be substracted from the positive terms.

emotion(horrible) = -5, capitalization rules -> emotion(horrible) = -6 emotion(happy) = 5

Decision value = -1

(S4) He is extremely brilliant but boring, boring.

Intensifier rule: the weight of extremely is 2, so the emotion(brilliant) = 4, emotion(boring) = -3, so the decision value is -2.

d)Specify the five elements of Bing Liu's model for Sentiment Analysis, and exemplify them with respect to the following text. Identify the features present in the text, and for each indicate its sentiment value as either positive or negative. Discuss two language processing challenges in automating the identification of such elements.

> "I am in love with my new Toshiba Portege z830-11j. With its i7 core processors, it is extremely fast. It is the lightest laptop I have ever had, weighting only 1 Kg. The SSD disk makes reading/writing operations very efficient. It is also very silent, the fan is hardly ever used. The only downside is the price: it is more expensive than any Mac. Lucia Specia, 10/04/2012."

Answer:

As for the five elements, they should be:

> `e_i` : the name of the entity
>
> `a_ij` : an aspect of the `e_i`
>
> `oo_ijkl` : the orientation of the opinion about the aspecrt `a_ij` of the entity.
>
> `h_k` : the opinion holder
>
> `t_l` : the time when opinion is expressedx by `h_k`

The name of the entity is `Toshiba Portege z830-11j` . For the many aspects, they should be `i7 core processors` , `SSD disk` , `weighting` , `fan` , `price` .The orientation of each opinion about the aspect, for the general idea about this product, the holders said `love` , it's the positive. For the aspect `i7 core processors` , the holder said `fast` , it is positive. For the `weight` , the holder said it's `the lightest` , the orientation is positive. As for the `SSD disk` , the holder said `efficient and silent` , so the positive. For the `fan` , it said `hardly ever used` , it's netural. The aspect `privr` , the `more expensive` , so it's negative. Aslo, you can generate the quintuple to show the result, like below:

(`Toshiba Portege z830-11j` , i7 core processors, positive,Lucia Specia,10/04/2012)

Challenge: 1) Maybe it's very difficult to comfirm the entity name because of variations and abbreviation. 2) It's very difficult to distinguish the subjective and objective sentences because of the language.

e) Differentiate direct from comparative Sentiment Analysis. What are the elements necessary in comparative models of Sentiment Analysis?

The comparative SA can contrast direct opinion verusu more comparativve opinions such as direct sentiment expressions on target objects and comparisons expressing similarities or differences.

h3

2016-2017

Section A

a)Explain briefly the intuition behind the PageRank algorithm. Discuss how it can documents that are ranked equally "relevant" according to the similarity score given by the vector space model.

PageRank is designed to use the structure of a graph to quantify the importance of each node in that graph. Accordingly, every usage of PageRank outside of a web context must maintain some notion of importance, even if the interpretation of the importance of a node varies from application to application.

PageRank can be thought of as a fluid that circulates throughout a network, passing from node to node across edges, and pooling at the nodes that are the most important

Intuition: PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

漫漫长途 技术为伴 © 2019 - 2021
Power by DoHerasYang | Jkeyll