Quantitative and qualitative computational analysis of language and text similarities, clustering and classification

Damir Ćavar
CLS 2010, August 2010
University of Zadar

Agenda

- General Aspects of Text Classification and Similarity Metrics
- Algorithms for Classification and Clustering
- Lexical Classification

Agenda

- General Intro
  - Documents and Data
    - Task
    - Feature Selection
      - Simple algorithms

Literature

- Coding:
  - Python
    - NLTK (book)
  - Scheme (Racket) (also the S-NLTK documentation)
  - R (CRAN)

Classification task

- See for an introduction:
  - Sebastiani (1999)
  - Manning, Raghavan & Schütze (2008)
  - Weiss, Indurkhya & Zhang (2010)

Text Representations

- List of tokens
  - may be filtered
  - may have frequencies or probabilities associated with every token
  - …
- List of features
  - Meta annotations

Document sources

- Corpora (static)
- Document streams (dynamic)
- Web (dynamic and noisy)
- Paper with Scanning and OCR
- etc.

Document standards

- Raw text (ASCII, Unicode)
- Structured (semantically annotated or not)
  - Tagged: HTML, XML, TEI, etc.
- Raw image oriented: PDF, TIFF, etc.

Text Classification

- Text
  - sequence of string tokens
    - words
    - symbols
    - format information
    - maybe grouped semantically

Goal 1

- Use text for automatic:
  - Classification (predefined set of labels) - supervised
  - Clustering (expected number of different groups) - unsupervised, exploratory

Classification task

- Given D ={D_1, D_2,…, D_n}, a set of *documents*
- Given C = {C_1, C_2,…, C_m}, a set of *classes* or *categories*
- find A = {$A_{1,1}$, $A_{1,2}$,…, $A_{n,m}$}, a set of *decisions*, with each element valued:

1. 0 or 1
2. $0 \leq A \leq 1$ (*fuzzy decision*)

Classification task

Think of a table representation, where the cells need to be filled:

	D_1	D_2	D_3	…
C_1	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	
C_2	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	
C_3	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	
…				

Classification task

- Notes:
  - Classes of categories are just *symbols*, their meaning is not related to the content (only maybe from the users perspective)
  - Elements of D are based on *content properties*, not specific *meta-data*

Classification task

- Inter-indexer inconsistency
  - Human subjects make inconsistent decisions wrt. the categorization of a document
    - Cleverdon (1984)

Classification task

- Variations of the task: Constraints
  - {≤1, 1, ≥1,…} elements of C are assigned to each element of D
  - If 1, then only one element of C is assigned to each D

→ *non-overlapping category constraint*

Classification task

- See for *non-overlapping category constraint*, e.g.:
  - Li & Jain (1988)
  - Schütze, Hull & Pedersen (1995)

Classification task

- Variations of the task: Constraints
- {≤1, 1, ≥1,…} elements of D are assigned to each element of C

Classification task

Processing strategies:

	D_1	D_2	D_3	…
C_1	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	
C_2	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	
C_3	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	
…				

- Category-pivoted categorization (classification) (CPC)
- Document-pivoted categorization (classification) (DPC)

Classification task

- ***Category-pivoted categorization*** **(CPC)**
  - *Row-based assignment* of each document to a category
  - Context and application:
    - The set D of documents is largely invariant
    - New categories are specified more frequently

Classification task

- ***Document-pivoted categorization*** **(DPC)**
  - Column-based assignment of a document to each category
  - Context or application:
    - Documents are processed sequentially or the category set C is largely invariant (Yang, 1999)
      - *On-line classification*
      - *Category-ranking classification*

Classification task

- DPC and CPC might be mixed in some scenarios
- The assignment values for DC pairs might be boolean decisions or fuzzy values (e.g. probabilities)
- Our scenario might be one class (boolean decision) per document (e.g. spam), or *n* classes per document

Applications

- Indexing in IR
  - keyword and key-phrase assignment
    - *controlled dictionary*
    - hierarchically organized dictionary: thesaurus (entries as classes)
- Approach: DPC, $1 \leq c \leq x$

Applications

- Document Management
  - Publishers
    - News
    - Company internal communication
- Common: 1 c for each d

Applications

- *Document filtering* or *document routing*
  - *dynamic* document collection (input *stream* with *producer* and *consumer*)
- Filtering: *relevant* and *irrelevant* documents
  - *non-overlapping categories*

Applications

- Search Space Categorization
  - Categorization of web pages

Applications

- *Word Sense Disambiguation* (WSD)
- Word senses as categories
- Context as documents

Feature Selection 1

Classification task

- Document/Content Features for classification:
  - How can we extract features?
  - How can we optimize the feature set?
  - What kind of feature sets do we need for what kind of algorithms?

Feature Generation

- First step:
  - Extract the content elements from documents
    - Do we use Meta-data?
- What are the content elements of documents?

Feature Generation

- Content encoded in language
  - words, sentences, text chunks
- Textual semantic properties
  - sections, chapters, titles, etc.
- Images, tables, numbers, etc.

Feature Generation

- Textual content features:
  - Break down the stream of characters, words and paragraphs into a list of single *tokens*
- **Tokenization**
  - Preserving the distributional relations between tokens

Feature Generation

- What are the problems in tokenization?
- Real example (CNN online):

Atlanta, Georgia (CNN) -- Former President Jimmy Carter is expected to arrive in the United States Friday with a U.S. citizen who was imprisoned in North Korea after entering it illegally in January, the Carter Center in Atlanta, Georgia, said.
The communist nation sentenced the American, Aijalon Mahli Gomes, to eight years of hard labor and a fine of about $600,000 for illegally crossing North Korea's border with China and for an unspecified "hostile act."

Tokenization

- What is desirable is a sequence representation:

Atlanta
,
Georgia
(
CNN
)
--
Former
President
Jimmy
Carter
is
expected
to
arrive
in
the
United
States
Friday
with
a
U.S.
citizen
who
...

Tokenization

- Identification of delimiters, words, sentences:
  - , . ; : - ! ? “ ‘ etc. (*visible*)
  - Space, Tab etc. (*invisible*)
- Delimiters or not:
  - 1,000.00
  - U.S.
  - etc.

Tokenization

- Problems:
  - Ambiguity of delimiters: e.g. . ’
  - Tokens with internal syntax and delimiters: e.g. 384-873, 27.5.2011, 1,000.-
  - Multi word tokens: e.g. John Doe, Dr. Smith, 22. Februar 2010
  - Language and orthography specific conventions require adaptation
  - Domain specific text requires adaptation

Tokenization

- Issues:
  - Orthography:
    - John reads the book.
    - The book is on the table.
    - THE END OF THE ECONOMIC CRISIS
  - Token difference: *The*, *the, THE*
    - upper/lower-case normalization

Source

- Simple implementation:
  - Code in Python: tokenizer.py
  - Weiss, Indurkhya & Zhang (2010), Page 18

First Approaches

Approaches

- Knowledge-engineering
  - Categorization decisions on the basis of manually defined rules, typically Boolean
    - if <condition> then <category>
- E.g. CONSTRUE

Approaches

- Simple rules might look like:
  - **if** *Hajduk* **in tokens**, **then** ***c*** **=** *sports*
  - **if** “Bologna program” **in tokens**, **then** ***c*** = *education*
  - **if** ***author*** **=** “Thomas Hanneforth”, **then** ***c*** **=** *Finite State Automata*

Approach

- Rules based on:
  - content properties
  - meta-data
  - etc.
- One or more rules per class
  - Voting

Approach

- Voting rules:
  - The *c* with the highest number of votes is assigned to a document.
  - All categories *c* with more than *x* votes are assigned to a document.
  - etc.

Approaches

- Problem: *Knowledge acquisition bottleneck*
  - Requires: knowledge engineer and domain expert
  - Manual changes and updates
  - Completely new design for every domain and document type

Approaches

- Alternative: Machine Learning Approach
  - Generation of an automatic generator of classifiers (induction), rather than generation of classifiers
  - Detection and generation of classes themselves

Approaches

- Common prerequisites:
  - Document corpus
    - set of documents with classifications assigned, which we can learn from
  - set of documents without labels
    - for the detection of document groups and potential document sets that might be assigned same class labels

Corpus

- Initial corpus *DC* and a set of categories *C*:

$$DC = \{d_1, d_2, \ldots, d_n\}$$

$$C = \{c_1, c_2, \ldots, c_m\}$$

- Division of *DC* in two subsets:
  - Training corpus: $DC_{tr} \subset DC$
  - Test corpus: $DC_{te} \subset DC$
  - and $DC_{te} \cap DC_{tr} = \emptyset$

Corpus

- Training and test corpus:

	Training corpus				Test corpus			
	D_1	D_2	…	D_n	D_{n+1}	D_{n+2}	…	D_m
C_1	$A_{1,1}$	$A_{1,2}$		$A_{1,n}$	$A_{1,n+1}$	$A_{1,n+2}$		$A_{1,m}$
C_2	$A_{2,1}$	$A_{2,2}$		$A_{2,n}$	$A_{2,n+1}$	$A_{2,n+2}$		$A_{2,m}$
C_3	$A_{3,1}$	$A_{3,2}$		$A_{3,n}$	$A_{3,n+1}$	$A_{3,n+2}$		$A_{3,m}$
…								

Corpus

- Splitting of training set:
  - *true* training set (or feature set) that leads to an optimized classifier
- *k*-fold Rotation over the corpus:
  - Select *k* different training and test sets from the corpus

Corpus

- Measuring generality of a category in a corpus *DC* (Shin et al. 2006):

$$g_{DC}(c_i) = \frac{|\{d_j \in DC | a_{ij} = 1\}|}{|DC|}$$

- and similar for the subsets: training and test corpus etc.

Evaluation

- *True positives*: the number of documents **correctly** assigned to a class
- *False positives*: the number of documents **wrongly** assigned to a class
- **Precision**

$$\frac{|\{\text{true positives}\}|}{|\{\text{true positives}\} \cup \{\text{false positives}\}|}$$

Evaluation

- Precision limits:
  - A score of 1.0 implies that all documents that were assigned to a certain class also belongs to this class
  - Ignores the number of documents belonging to class C that were not assigned to it

Evaluation

- *False negatives*: the documents that belong to some class c_i, but have not been labeled as such
- **Recall**

$$\frac{|\{\text{true positives}\}|}{|\{\text{true positives}\} \cup \{\text{false negatives}\}|}$$

Evaluation

- Recall limits:
  - A score of 1.0 means that all documents belonging to class C have been labeled appropriately
  - Ignores the number of documents wrongly assigned to class C

Evaluation

- **F-measure**
  - Harmonic mean of precision and recall

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Code Examples

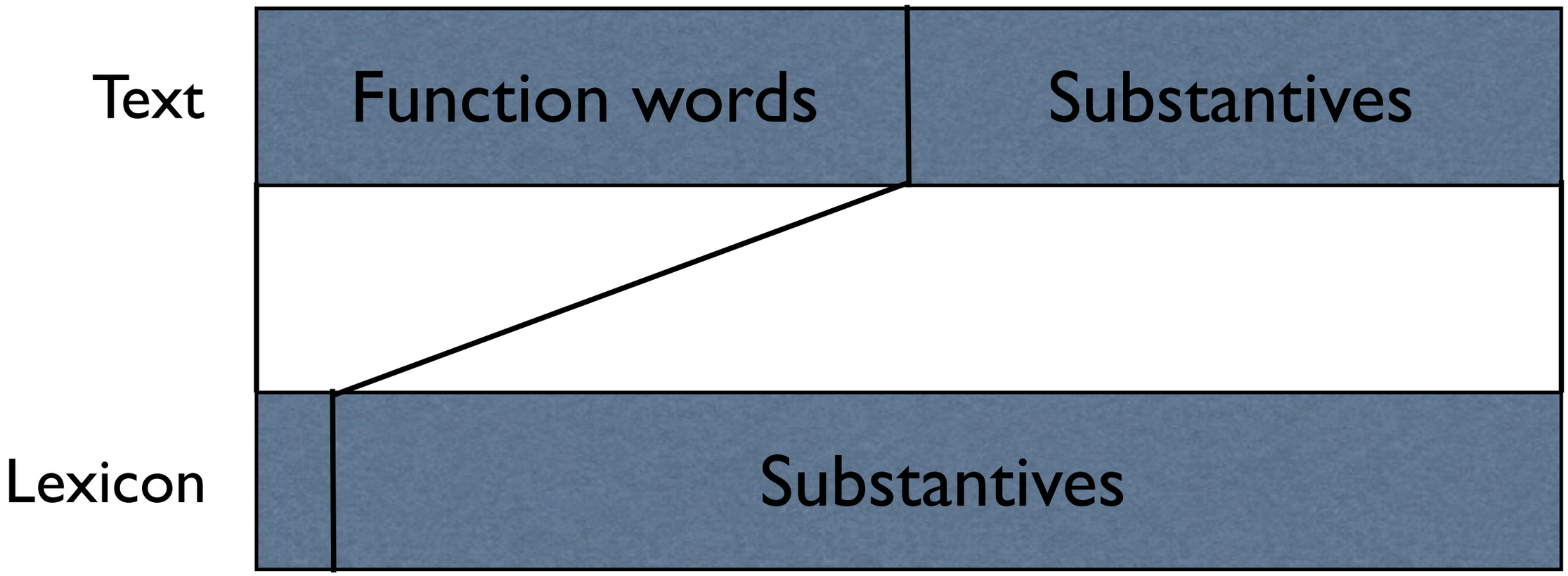
- Python: NLTK metrics module
- Perl: Statistics::Contingency
- R: precision.recall
- Scheme: S-NLTK ?
- and many others...

Feature Selection 2

Simple Comparison

- Type/Token Ratio (e.g. Williamson 2009)
  - Variation of vocabulary
    - Child-oriented speech vs. adult conversations
    - Newspaper articles vs. fiction or scientific papers

Type Token Relations



Type/Token Ratio

- See example: ttr.py
  - as parameter: token list (as text file)
  - lower-case normalization of all tokens

Frequency Profiles

- Frequencies of tokens in text
  - Highly frequent: function words
  - More than n ($n \geq 2$) times: Common substantives
  - Low frequent: specific substantives
  - In specific texts there will be some specific distribution properties of common substantives and usually low frequent terms

Frequency Profiles

- Example: fp.py and rfp.py
- Data1.xlsx

Unique Tokens

- Filtering unique tokens per class
- Example: unique.py

Frequency Profiles

- Frequencies of tokens in individual documents: Relative Frequency
  - rfp.py
- Unique word frequencies in individual documents:
  - rfp-unique.py

Frequency Comparison

- Calculate the distance between the models:
  - First naive idea:
    - For any unknown document:
      - Sum the distance between the distributional properties of all the tokens to all the frequency profiles

Example

- Python: unknown-class1.py
- Python: Language Identification
  - lid.py & lidtrainer.py

Frequency Profiles

- Unigram, bigram or digram, trigram... models: N-gram models
  - sequence of length 1 to N of tokens, characters, etc. with associated probability
- What do we just ignore?

Frequencies and N-gram models

Entropy

- If we want to encode *n* symbols (strings, characters, etc.) in a binary system, how many bits do we need to encode them?
  - 1 bit: 0 or 1 = 2 symbols
  - 2 bits: 00, 01, 10, 11 = 4 symbols

Entropy

- In general, for *n* symbols we are looking for x such that:

$$2^x = n$$

- For 3 symbols we need bits:

$$x = log_2(3) \approx 1.5849625$$

- We take the ceiling for whole bits:

$$x = \lceil log_2(3) \rceil = 2$$

Entropy

- The negative *log* of the reciprocal of an integer is equivalent to the *log* of the integer:

$$x = log_2(3) = -log_2\left(\frac{1}{3}\right) \approx 1.5849625$$

- For each individual of our 3 symbols, each equal likelihood:

$$-\frac{1}{3}log_2\left(\frac{1}{3}\right)$$

Entropy

- In general for *n* symbols with equal likelihood, we need *x* bits:

$$\left\lceil -\sum_{i=1}^{n} \frac{1}{n} log_2 \left(\frac{1}{n} \right) \right\rceil$$

- Example:
  - $S = \{a, b, c, d\}$, and it is equally likely for each $s \in S$ to occur, i.e. $p(s) = 1/4$, how many bits do we need?

Entropy

- For *n* symbols in S = {a, b, ..., x}, and for each s ∈ S, *p*(*s*) = 1/*n*:

$$x = -\sum_{i=1}^{n} \frac{1}{n} log_2 \left(\frac{1}{n}\right)$$

$$= -\sum_{i=1}^{n} p(s_i) log_2 \left(p(s_i)\right)$$

- If S = {a, b, c, d}, and p(a) = p(b) = p(c) = p(d) = 1/4 = 0.25:

$$-\sum_{i=1}^{n} \frac{1}{4} log_2 \left(\frac{1}{4}\right) = 2$$

Entropy

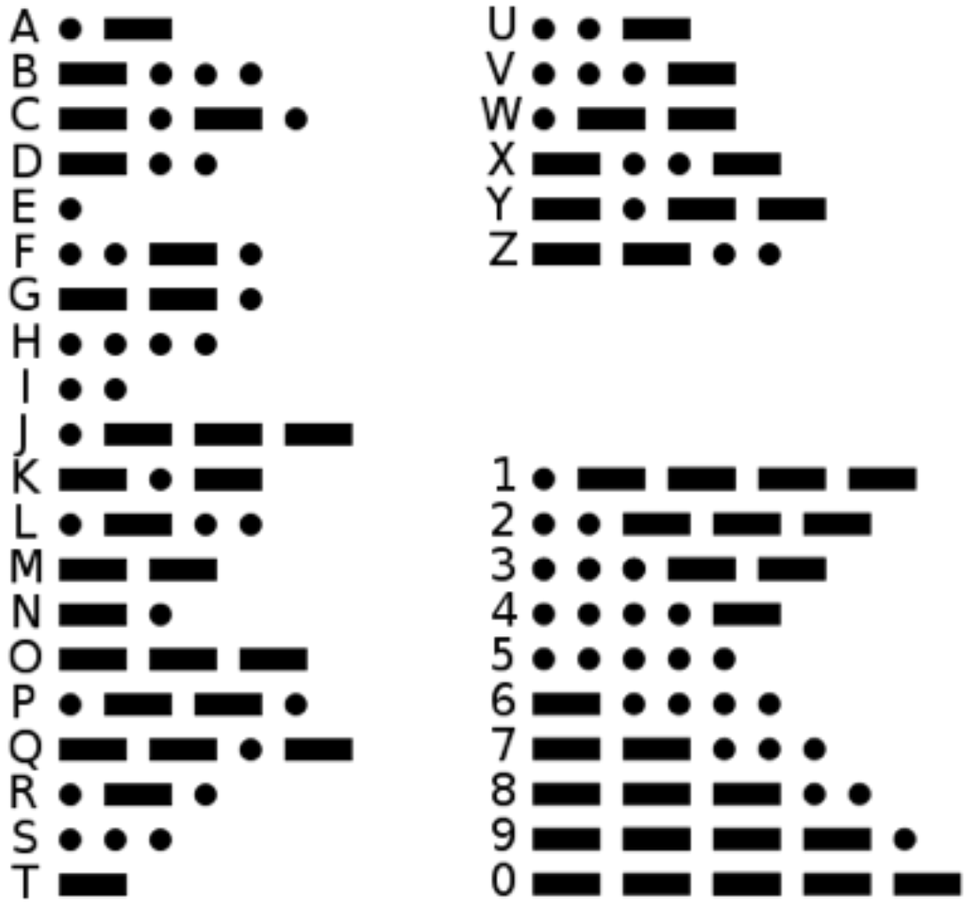
- If e.g. symbols in text occur with different probabilities:
  - we can optimize the binary code for symbols by assigning less bits to more frequent symbols
- Image from MacKay (2003)

i	a_i	p_i
1	a	0.0575
2	b	0.0128
3	c	0.0263
4	d	0.0285
5	e	0.0913
6	f	0.0173
7	g	0.0133
8	h	0.0313
9	i	0.0599
10	j	0.0006
11	k	0.0084
12	l	0.0335
13	m	0.0235
14	n	0.0596
15	o	0.0689
16	p	0.0192
17	q	0.0008
18	r	0.0508
19	s	0.0567
20	t	0.0706
21	u	0.0334
22	v	0.0069
23	w	0.0119
24	x	0.0073
25	y	0.0164
26	z	0.0007
27	–	0.1928

Entropy

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.



Entropy

- We can compare frequency tables of symbols (characters, words, phrases, concepts etc.)
  - Croatian frequency profiles of characters, words, phrases are different from English ones.
  - Frequency profiles of different text types and genres in the same language are maybe different.

Entropy

- With $S = \{a, b, c, d\}$
- and $P_s = \{0.4, 0.3, 0.1, 0.2\}$

- Note: $\sum_{i=1}^{n} p(s_i) = 1$

- Bits required: $-\sum_{i=1}^{n} p(s_i) log_2(p(s_i))$

Entropy

- Uncertainty of obtaining some outcome s_x from the S
- Compare:

$$\{S : a = 0.25, b = 0.25, c = 0.25, d = 0.25\}$$

$$\{S : a = 0.2, b = 0.3, c = 0.25, d = 0.25\}$$

$$\{S : a = 0.01, b = 0.97, c = 0.01, d = 0.01\}$$

- How do the entropy overall, and the uncertainty of getting an outcome *b* in particular differ?

Entropy

- See code example: entropy1.py

$$\{S : a = 0.25, b = 0.25, c = 0.25, d = 0.25\} = 2$$

$$\{S : a = 0.2, b = 0.3, c = 0.25, d = 0.25\} \approx 1.98547529723$$

$$\{S : a = 0.01, b = 0.97, c = 0.01, d = 0.01\} \approx 0.241940732853$$

Point-wise Entropy

- See code example: pentropy1.py
  - What is the uncertainty to obtain a *b*?

{a:0.25, b:0.25, c:0.25, d: 0.25}
a 0.5, b 0.5, c 0.5, d 0.5

{a:0.1, b:0.5, c:0.2, d:0.2}
a 0.332192809489
b 0.5
c 0.464385618977
d 0.464385618977

{a:0.01, b:0.97, c:0.01, d:0.01}
a 0.066438561897 7
b 0.0426250471 6,
c 0.066438561897 7
d 0.066438561897 7

Use

- Comparison of sizes of symbol distributions in bits
  - Class-Keyword relations
    - a class is represented by different distributions of keywords (tokens, phrases)
    - for a given distribution, within each class the number of bits might be different
    - assign to a distribution of symbols the class that would require the smallest encoding

Entropy in Real Life?

- Truncation and reduction phenomena in spoken language?
  - Thomas -> Tom
  - is -> s
  - etc.

N-gram models

Feature Selection

- List of tokens
  - *Peter reads a book . John and Mary read some newspaper .*
  - Two tokens with the same meaning, type etc.: *reads*, *read*
  - Normalization via lemmatization
    - *reads* → *read*