2 The Information Retrieval data model

2.1 Introduction: Data models

2.2 Text in BioSciences

2.3 Information Retrieval Models

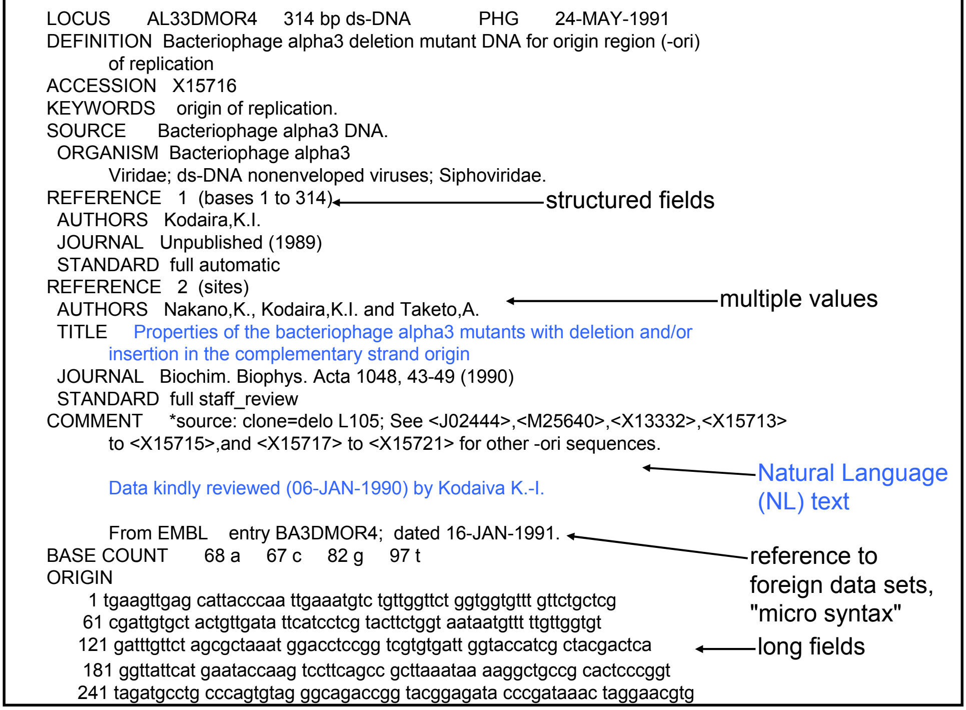
2.3.1. Boolean Model

2.3.2 Vector space Model

2.3.3 Efficient implementation of VSM

2.4 Evaluation of Retrieval effectiveness

2.5 Molecular Biology application



```
LOCUS       AL33DMOR4     314 bp ds-DNA           PHG       24-MAY-1991
DEFINITION  Bacteriophage alpha3 deletion mutant DNA for origin region (-ori)
            of replication
ACCESSION   X15716
KEYWORDS    origin of replication.
SOURCE      Bacteriophage alpha3 DNA.
  ORGANISM  Bacteriophage alpha3
            Viridae; ds-DNA nonenveloped viruses; Siphoviridae.
REFERENCE   1  (bases 1 to 314)
  AUTHORS   Kodaira,K.I.
  JOURNAL   Unpublished (1989)
  STANDARD  full automatic
REFERENCE   2  (sites)
  AUTHORS   Nakano,K., Kodaira,K.I. and Taketo,A.
  TITLE     Properties of the bacteriophage alpha3 mutants with deletion and/or
            insertion in the complementary strand origin
  JOURNAL   Biochim. Biophys. Acta 1048, 43-49 (1990)
  STANDARD  full staff_review
COMMENT     *source: clone=delo L105; See <J02444>,<M25640>,<X13332>,<X15713>
            to <X15715>,and <X15717> to <X15721> for other -ori sequences.

            Data kindly reviewed (06-JAN-1990) by Kodaiva K.-I.

            From EMBL    entry BA3DMOR4;  dated 16-JAN-1991.
BASE COUNT       68 a     67 c     82 g     97 t
ORIGIN
        1 tgaagttgag cattacccaa ttgaaatgtc tgttggttct ggtggtgttt gttctgctcg
       61 cgattgtgct actgttgata ttcatcctcg tacttctggt aataatgttt ttgttggtgt
      121 gatttgttct agcgctaaat ggacctccgg tcgtgtgatt ggtaccatcg ctacgactca
      181 ggttattcat gaataccaag tccttcagcc gcttaaataa aaggctgccg cactcccggt
      241 tagatgcctg cccagtgtag ggcagaccgg tacggagata cccgataaac taggaacgtg
```

structured fields
multiple values
Natural Language (NL) text
reference to foreign data sets, "micro syntax"
long fields

2.1 Introduction: Datamodels

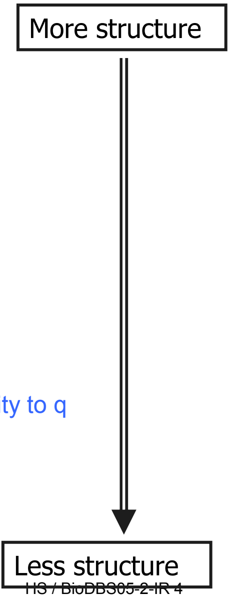
- What is appropriate for bio data?
  - No clear answer
  - Ideological positions not helpful
  - adaption of new techniques makes sense – but takes a long time - i.g. ASN.1 ⇨ XML, relational ⇨ Oo (?)
- Pragmatic requirements
  - Flexibility
  - ease of use
  - few restrictions

… new types of objects, change of identifiers, …, …everything resulting from progress of science

Anything more flexible than natural language?

Data models: the spectrum

- Databases
  - Rigid data models: relational, object-oriented
  - Database conformant to schema
  - Semantics of query q: subset of database
  - No texts, images, ... (oríginally)
- Semi structured DB / XML
  - Schema more flexible – if any
  - Many schema items
  - Text plays a big role
  - Semantics of queries: substructure of DB
- Information Retrieval
  - Data model: objects are sequences of terms
  - No modeling restrictions (natural language!?)
  - Semantics of query q: DB entries ordered by similarity to q (Ranking)
- Natural Language
  - VERY difficult to process automatically ⇨ not really an option



More structure

Less structure

2.2 Text in BioInformatics

- Automatic processing of text

Use statistical methods and heuristics for simple tasks!

Example: finding abbreviations

"Abbreviation mining" (Chang, Schütze, Altmann, Stanford)

Method: text alignment using dynamic programming

MEDLINE: "According to a system proposed by the European group for the immunological classification of leukemia (EGIL)….."

```
European group for the immunological classification of leukemia
E........G.............I...............................L.......
```

Evaluation

- Abbreviation server: http://abbreviation.stanford.edu/

local1 local2

Evaluation

37 GB Medline, 452 entries of an lexikon of abbreviations

375 out of 452 found correctly ("Recall")

402 of 452 classified correctly ("Precision")

Details see: Chang et al: Creating an online dictionary of Abbreviations from Medline, Journal of the American Medical Informatics Association Volume 9 Number 6 Nov / Dec 2002

Information Retrieval

- … works on textual data
- Why useful in BioSciences
  - "Most data (knowledge?) buried in text" - journal paper, proceedings, databases…

  ⇨ Standard retrieval task as in other diciplines

  … did you know that the number of scientific papers doubles every ~12 years

  - Information Retrieval techniques may be (are?) useful for similarity search in nucleotide sequences
    - comparison of a query string to EACH sequence in the DB takes time
    - text indexing techniques may help

2.3 Information Retrieval models

- Document ("data") model

D = "set of documents",

K = {k_1,...,k_n} set of index terms

K ~ set of all words occurring in the database

Typically very large

For every dj ∈ D, k_i ∈ K there is a weight $w_{ij} \geq 0, w_{ij} \in$ *Real*,

if k_i does not occur in d_j => $w_{ij} = 0$

dj'= $(w_{1j}, ...w_{nj})$ is the document representation of dj

identify d_j' and d_j in most cases, i.e. D = {d_j | d_j = $(w_{ij}, ...w_{nj})$ }

i.e. a document is a high dimensional vector of real numbers, most of them are 0, each component represents a term ∈ K.

2.3.1 Boolean retrieval

- Model
  - $w_{ij} = 1$ if term k_i occurs in document d_j, else 0
  - Query language: boolean expression of $k_i \in K$
  - Evaluation of a query q:

    let $d_j \in D$ a document vector of 0 and 1,
     - if q = ki then d matches q iff $d_{ij} = 1$
     - if q = "q1 AND qj " q matches dij if q1 matches d_{ij} and
       q2 matches d_{ij}
     - if q = "q1 OR qj " q matches dij if q1 matches d_{ij} or
       q2 matches d_{ij}
     - if q = "NOT q1" q matches dij if q1 does not match d_{ij}

- Implementation
  - Conceptually simple
  - Efficient query evaluation
  - many library systems / online retrieval systems work use it

Boolean retrieval

- Issues
  - Very restrictive evaluation: binary decision

    Wanted: mapping s: Q x D -> [0,1]

    Q is the set of all queries
  - Every term has the same influence on the result

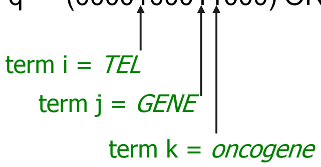
    Wanted: weight should reflect "importance" of term

    Example:
    term "protein" occurs in many   documents
    many times, term "propylthiouracil" less frequent….
    In a search for "propylthiouracil AND protein" both have
    the same significance.
  - For q = "k_1 OR…. k_j" a document matches if
    at least one term matches.
    No difference if one or all terms match.

Boolean retrieval

- Coordinate level match
  - Let q be in disjunctive normal form:
    $q' = \underset{i}{\mathrm{DISJ}} (t_{i1} \text{ AND } t_{i2} \text{ AND} \ldots\text{AND } t_{ik}),\ t_{ij} = 0 \text{ or } 1$

    Example: q =(*TEL* or *gene 6*) and *oncogene*
    q' = (111) OR (101) OR (011)
  - Extend each disjunctive term by 0's for all terms in K not occuring in q
  - q' = (0000100011000) OR … = $qSig_1$ OR…OR $qSig_k$



term i = *TEL*

term j = *GENE*

term k = *oncogene*

$$\boxed{s(q,d) = \max_i (qSig_i * d) \quad (* : \text{scalar product})}$$

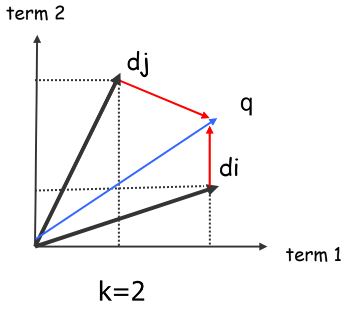
Means: the more query terms found in document, the better

Boolean Retrieval

- Discussion
  - (+) Ranking
  - (+) number of matching query terms in document d define rank of d
  - (-) Rank dependent on number of query terms
  - (-) Documents with many terms tend to be ranked higher
  - (-) Terms which occur frequently in documents are treated in the same way as infrequent terms
- Requirement
  - More general term weights
  - Normalization of ranking s(q, dj)

2.3.2 Vector space model

- Model
  - Documents: points in a |K| = n – dimensional vector space.
  - Weights normalized e.g. 0 <= w <= 1
  - Terms are independent of each other ("orthogonal")



  - Queries …..

    …. are (formally) documents: q= (q1, q2, …,qn)

- Needed: measure of similarity between document and query, e.g. vector difference.

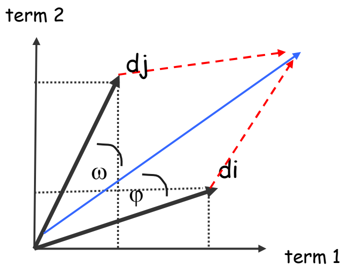
Vector space: similarity function

- Heuristic similarity functions
  - Scalar product?
    w1j*q1 + w2j*q2 + ... + wnj*qn
    not bounded, may become arbitrarily large
  - Cosine measure

    Cos (dj,q) = cos φ

    = dj • q / |dj| * |q|

    $= \sum wij*qi / \sqrt{} (\sum wij^2) * \sqrt{} (\sum qi^2)$



Measures angle between query vector and document and normalize.

Weights

- How to assign weights to documents / queries
  - Manual weight? Impossible! (more or less)
  - wanted: dj = (w_{1j}, ..., w_{nj})

- Document frequency
  - Remember: infrequent terms are typically more significant
    than frequent ones
    "protein" compared to "interleukin 3"
  - Hypothesis: importance of a term depends on number of documents it occurs in
  - Justification: Zipf's law
    Frequency of an event is inversely proportional to its significance
    (Human Behaviour and the Principle of Least effort (G. Zipf 1949))

Weights

… Zipf's law

example *(see http://information-retrieval.de/irb)*

– Consistent to information theory (Shannon) -->

⇒
Weight w of term t inverse proportional to document frequency

Document frequency DF of term t:
the number of documents, term t occurs in

Term frequency TF of term t in d :
number of occurences of t within one document d

Information Theory

- Huffman – Code

  Given an alphabet A = {a1,....,an} and probabilities of occurrence pi = p(ai) in a text for each ai.

  Find a binary code for A which minimizes
  $H'(A) = \Sigma$ pi * length (cw_i), cw_i = binary codeword of ai

  H'(A) is minimized for $\text{length}(cw_i) = \lceil \log_2 1/ \text{pi} \rceil$

  well known how to construct it... ⇨ intro to algorithms

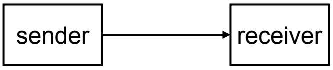
  $H(A) = - \Sigma$ pi * $\log_2$ pi : important characterization of A
  what does it mean?

Entropy: interpretations

- Entropy

  $H(A) = - \Sigma$ pi * $\log_2$ pi

  – minimal number of bits to encode A



sender

receiver

  – amount of uncertainty of receiver before seeing an event (a character transmitted)

  – amount of surprise when seeing the event

  – the amount of information gained after receiving the event.

Information Theory and alphabets

- Example

  $L = \{A,C,T,G\}$, $p(A) = p(C) = p(T) = p(G) = ¼$,

  Boring: seeing a "T" in a sequence is as interesting as seeing a "G" or seeing an "A".

  $H(L) = - ¼ * \Sigma \log 1 – \log 4 = 2$

  But:

  $L' = \{A,C,T,G\}$, $p(A) = 0.7$, $p(C) = 0.2$, $p(T)= p(G) = 0.05$

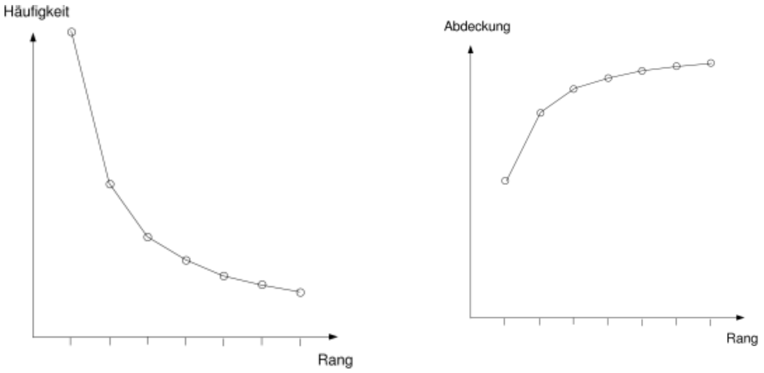
  Seeing a "T" or a "G" is exciting as opposed to "A"

  $H(L') = -(-0.7*0,514 – 0.2*2.31- 2* 0.05*4.32)$
  $= 0.36 + 0.464 + 0.432 = 1.256$

  Low entropy more interesting

  What is the lowest value?

Zipf's law



Zipf's law applied to text documents: Frequency and cover of text

from Ferber: Information Retrieval
http://information-retrieval.de/irb/ir.part_1.html
y-coord??

Weights

- Term frequency

  Base hypothesis of Information Retrieval:

  1. Frequency of term t   TF   characterizes contents of document j
     TF(t,j) =: $f_{t,j}$

  2. Document frequency characterizes term: DF(t) := f_t

inverse document frequency IDF = 1/ f_t used in similarity functions

TF / IDF

- Normalization
  – TF should not be linear   (…why?)

    normalization heuristics ⇨ normalized term frequency $r_{t,j}$

    e.g. $r_{tj} = 1 + \log f_{tj}$

    or $r_{tj} = k + (1-k)\ \ f_{tj} / \max_i f_{ij,,}$some constant $0 < k \leq 1$

  – IDF should be independent of number of documents
    - normalization heuristics ⇨ weight w_t:

      e.g. $w_t = \log (1 + N/f_t)$, N = number of documents

      or $w_t = \log (1 + f_{max} / f_t)$

  … many other heuristics

Weights

- Cumulative weight of term t in document j

$w_{t,j}$ = g (TF, 1/DF) = g (TF, IDF) , some function g

Weight of term t in document dj ("TF / IDF heuristics")

Typical: $w_{tj} = r_{tj} * w_t$

i.e.   dj = (w_{1j}, ..., w_{nj})

- Weight of a query term

$w_{tq} = q_t * w_t$, ,

q_t = weight relative to query.

Typical: $q_t = 1$ ("All terms equally important")

Ranking

Calculating similarity of query q and document d_j using cosinus measure

Document dj sometimes abreviated as "

Cos(dj, q)

$$= dj \bullet q \;/\; |dj| * |q|$$

$$= \sum_{t \in dj \cap q} w_{tj} {*} w_t \;/\; \sqrt{} (\sum w_{tj}{}^2) * \sqrt{} (\sum w_{tq}{}^2)$$

$$= 1/ (Wj{*}Wq) \;\; * \sum (1 + \log f_{t,j}) *(\log (1 + N/f_t))^2$$

where $Wj = \sqrt{} (\sum w_{t,j}{}^2)$, $Wq = \sqrt{} (\sum w_{t,q}{}^2) = \sqrt{} (\sum w_t{}^2)$

Note: document frequency has double influence - counts in d_j as well as q. Reasonable?

Ranking

Most often used for ranking of document / query similarity:

$$\mathrm{Cos}(dj,q) = 1/\,(Wj{*}Wq)\ {*} \sum_{t \in dj \cap q} (1 + \log f_{t,j}) * \log(1 + N/f_t)$$

Rank for each document in document set D
==> Ranking of result set

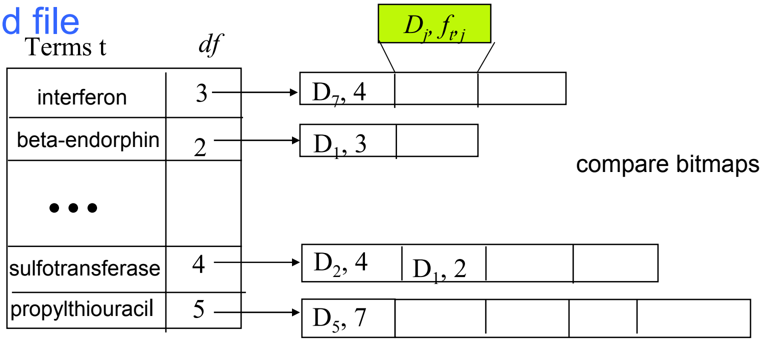
Issues:

- Efficient implementation
- Evaluation of "retrieval effectiveness"

- Many more similarity measures.
- Specific measures for Web documents (e.g. Google: "page rank")
- Domain specific measures

2.3.3 Implementation of vector space model

Inverted file



Terms t	*df*
interferon	3
beta-endorphin	2
• • •	
sulfotransferase	4
propylthiouracil	5

compare bitmaps

Index file,
e.g. B+ tree,
suffix tree

Postings lists,
suited for boolean queries
and similarity search

Most values can be calculated before processing a query and put into the posting list, e.g. 1+log f_{tj}

2.4 Evaluation : recall / precision

- Issues
  - Subjectiveness of judgement

    How relevant is a document with respect to a query?
  - Elaborate, costly empirical tests required

    many queries, many individual judgements

    for each query, mean of judgements?
- Evaluation model
  - Ideal observer: knows relevant documents for each query
  - Check for each query q
    - how many relevant documents found
    - how many irrelevant documents found
  - Calculate mean over many queries

Evaluation

	relevant	not relevant
found	r	n
not found	v	u

Recall:

fraction of *relevant* documents *found* out of all *relevant* documents

$R = r / (r + v)$

Precision:

fraction of *relevant* documents *found* out of *all* documents *found*

$P = r /(r + n)$

Noise: ….?

F-Measure:

$F = 2P*R / P+R$

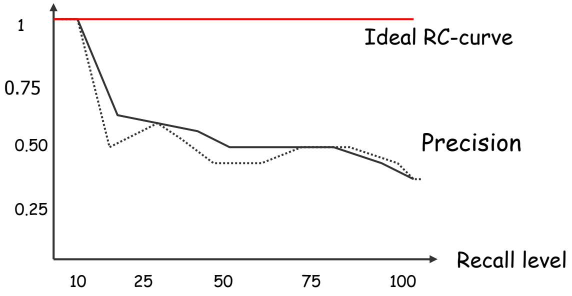
Relevant objects found should occur as soon as possible in Output set.

How to evaluate ranking *order*?

Evaluation

	Recall level	Precision %
1 ✓	10	100
2	10	50
3	10	33
4 ✓	20	50
5 ✓	30	60
6	30	50
7 ✓	40	57
8	40	50
9	40	44
10	40	40
11	40	36
12 ✓	50	42
13 ✓	60	46
14 ✓	70	50
15	70	47
16 ✓	80	50
17	80	47
18	80	44
19 ✓	90	47
20	90	45
21	90	43
22 ✓	100	45
23	100	43
24	100	42
25	100	40

Recall-Precision Graph



Recall level n:
n % of all relevant
Documents have
been found
for a particular query and document set!

RC curve:
Precision at
recall level n

2.5 Molecular biology applications

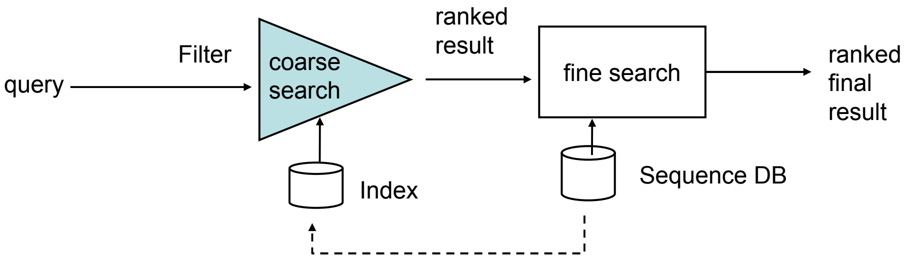
- TREC competition
  - – on retrieval of publications in MEDLINE concerning the function of a gene
    - ⇨ GeneRIF ("reference into function")

Query example:
For gene X, find all MEDLINE references that focus on the basic biology of the gene or its protein products from the designated organism. Basic biology includes isolation, structure, genetics and function of genes/proteins in normal and disease states.

details see paper

Molecular biology applications

Does indexing help to reduce search time for similar sequences ?



see: Aung et al: Rapid 3D protein structure Database searching using information retrieval techniques, Bioinformatics 20(7), 2004 pp1045-1052

see: William, Zobel: Indexing and Retrieval for Genomic Databases, 2002

Indexing sequences

- Intervals

Insulin - Goose
AANQHLCGSHLVEALYLVCGERGFFYSPKT/GIVEQCCENPCSLYQLENYCN*
>P1;INAQ
Insulin - American alligator
AANQRLCGSHLVDALYLVCGERGFFYSPKG/GIVEQCCHNTCSLYQLENYCN*
… …

n length of each interval (n –grams) ,
k length of sequence ⇨ k-n+1 intervals

Example: acctgtc , n=3 : acc, cct, ctg, tgt, and gtc.

Construct inverted index for interval occurrence in the sequences of DB

correspondence: Document ⇔ sequence, word (term) ⇔ interval

Inverted genomic index

Posting list:

<interval> [seqNo, (noOfMatches: positions)]*

e.g. GAGA -> 52,(3: 147,233,256), 83,(2: 17, 256)
means: GAGA occurs 2 times in sequence 52 at positions 147, 233, 256 and 2 times in 83…

Coarse homology search of sequence s in the DB:
- Cut s into intervals
  - locate matching sequences in DB and the matches of intervals of s
  - rank the matching sequences
  - fine search (e.g. alignment) on sequences

Ranking

"Frame-based" ranking (Williams / Zobel),

Frame= set of matching intervals with the same offset

```
A        10        20        30       40        50
ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
        ::::              ::::::::
  GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
            10       20        30
B        10        20         30       40       50
ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
                ::::
 GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
        10        20         30
C        10        20         30       40        50
ACCCTGAGGTTTTTTTTGGGAGAGCTTTCTTCTTAGAGAGGAGGCTAGCTAGCTTCG
                                                    ::::
                          GTGTGTGTTTGTGTGTGGGGTAAGTTCTTCTTCTT
                                    10        20         30
```

Frame F2 of A and search sequence: matching offset for interval GTT: 2
Matches of F2 with offset 2: (9,7), (10,8),(27,25),(28,26),(29,27),(30,28),(31,29), (32,30)

Frame F26 of C and search sequence: matching offset for interval GTT: 2
Matches of F2 with offset 26: (53,27), (54,28)

Ranking

Frame F2 of A and search sequence: matching offset for interval GTTT: 2
Frame with offset 2: (9,7), (10,8),(27,25),(28,26),(29,27),(30,28),(31,29), (32,30)

Frame F26 of C and search sequence: matching offset for interval GTTT: 2
Frame with offset 26: (53,27), (54,28)

Simple ranking scheme :
Rank s, t according to number of intervals in each frame for s and t.*

rank(s,t) = max (|F (I(s) $\cap$ I(t)) |) I(x) = number of intervals in x, F(): returns sets of intervals at the same offset.

How effective? Time, precision, recall, storage space?

* May be improved, not important in this context, see paper

Evaluation: P/R Graph

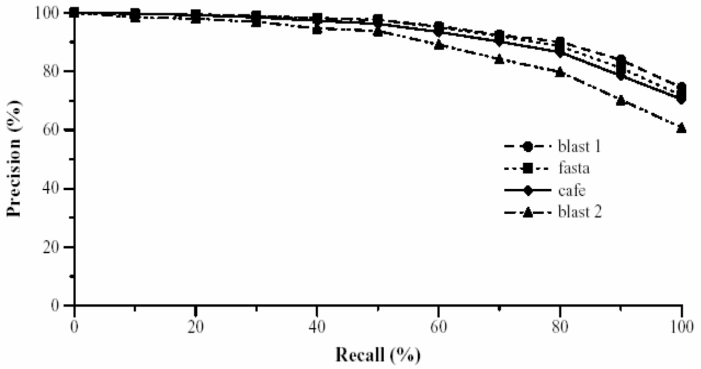


Figure 8: *Mean recall-precision for both versions of* BLAST, FASTA, *and* CAFE. *The* PIRSF *collection is used, with 1,834 amino-acid queries. We parameterise* CAFE *to use a banded local alignment fine search (similar to that used in* FASTA*) and the* NEIGHBOURHOOD *frames ranking metric. All systems use a PAM-250 matrix scoring matrix.*

cf. from Williams / Zobel

Evaluation: Time

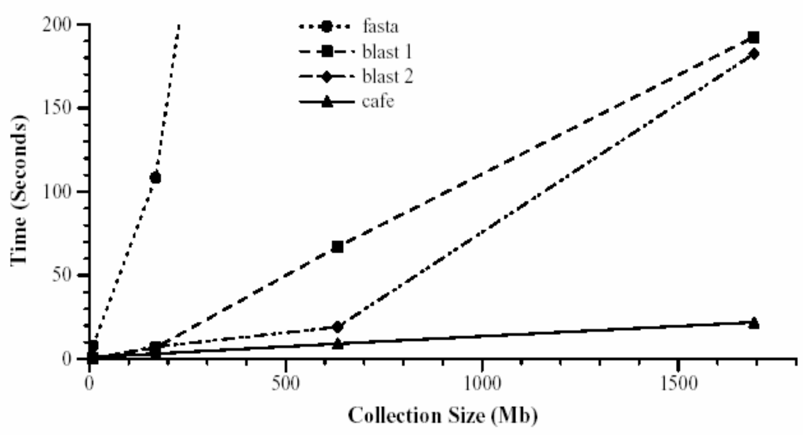
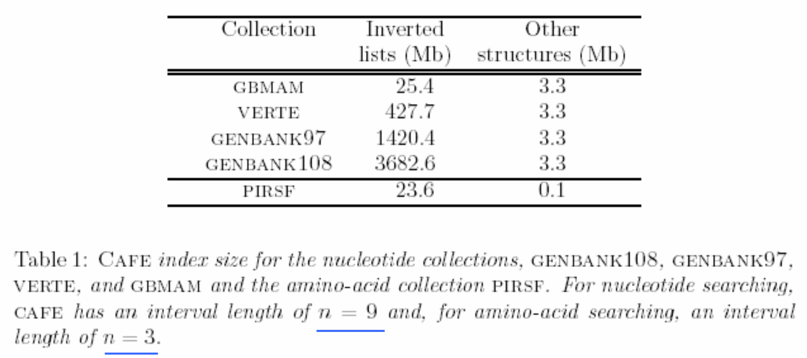


Figure 9: *Plot of the time for* BLAST, CAFE, *and* FASTA *to search the nucleotide collections* GBMAM, VERTE, GENBANK97, *and* GENBANK108, *averaged over 41 queries.*

cf. from Williams / Zobel

Evaluation: Space



Collection	Inverted lists (Mb)	Other structures (Mb)
GBMAM	25.4	3.3
VERTE	427.7	3.3
GENBANK97	1420.4	3.3
GENBANK108	3682.6	3.3
PIRSF	23.6	0.1

Table 1: CAFE *index size for the nucleotide collections,* GENBANK108, GENBANK97, VERTE, *and* GBMAM *and the amino-acid collection* PIRSF. *For nucleotide searching,* CAFE *has an interval length of $n = 9$ and, for amino-acid searching, an interval length of $n = 3$.*

Index size

Genbank: factor of 2.2 compressed

factor > 10 uncompressed

Summary

- Information retrieval:
  - essential for text retrieval
  - in molecular biology in particular
    - who published the data when? which experiments?...
  - Indexing seems to be an alternative for a coarse searching step
  - Ranking functions: domain specific heuristics