Learning in Information Retrieval (1)

Outline

Learning in IR – introduction/overview

Some applications

Case Study 1: Genetic Algorithms in IR

Case Study 2: Genetic Programming in IR

Available Software

Summary

Motivations

Many real-world problems are complex and it is difficult to specify (algorithmically) how to solve many of these problems.

Learning techniques are used in many domains to find solutions to problems that may not be obvious/clear to human users.

"How can computers learn to solve problems without being explicitly programmed?" A. Samuel (1959)

Many applications in IR including …

Classification

Clustering

Neural network models

Learning user models

Learning optimal means to combine sources

Learning to rank

and many others …

In general …

Machine learning involves searching a large space of potential hypotheses or potential solutions …

… to find the hypotheses/solution that best *explains* or *fits*

  a set of data and

  any prior knowledge

or is the best solution.

or can say … *learns if it improves its performance*.

Machine learning techniques require a training stage before the learned solution can be used on new previously unseen data .

The training stage consists of a data set of examples which can be:

- Labelled (supervised training)
- Unlabelled (unsupervised training)

An additional data set must also be used to test the hypothesis/solution.

Symbolic

knowledge is represented in the form of symbolic descriptions of the learned concepts, e.g., production rules or concept hierarchies

Sub-symbolic

Knowledge is represented in a sub-symbolic form not readable by a user, e.g., in the structure, weights and biases of the trained network

Learning approaches have become more and more prevalent in information retrieval in the past few years.

-New workshops, conference, application domains, systems/packages, approaches

-“Learning to Rank”

Many examples of learning in IR:

Feedback mechanisms; clustering;

Sentiment analysis using neural networks

(Santos & Gatti, 2014)

A Language Modeling Approach to Personalized Search Based on Users’ Microblog Behavior (Younus et al, 2013)

Learning to Extract Local Events from the Web

(Foley et at., 2015)

Learning to Reweight Terms with Distributional Representations

(Zheng, Callan, 2015)

... and many many more ..

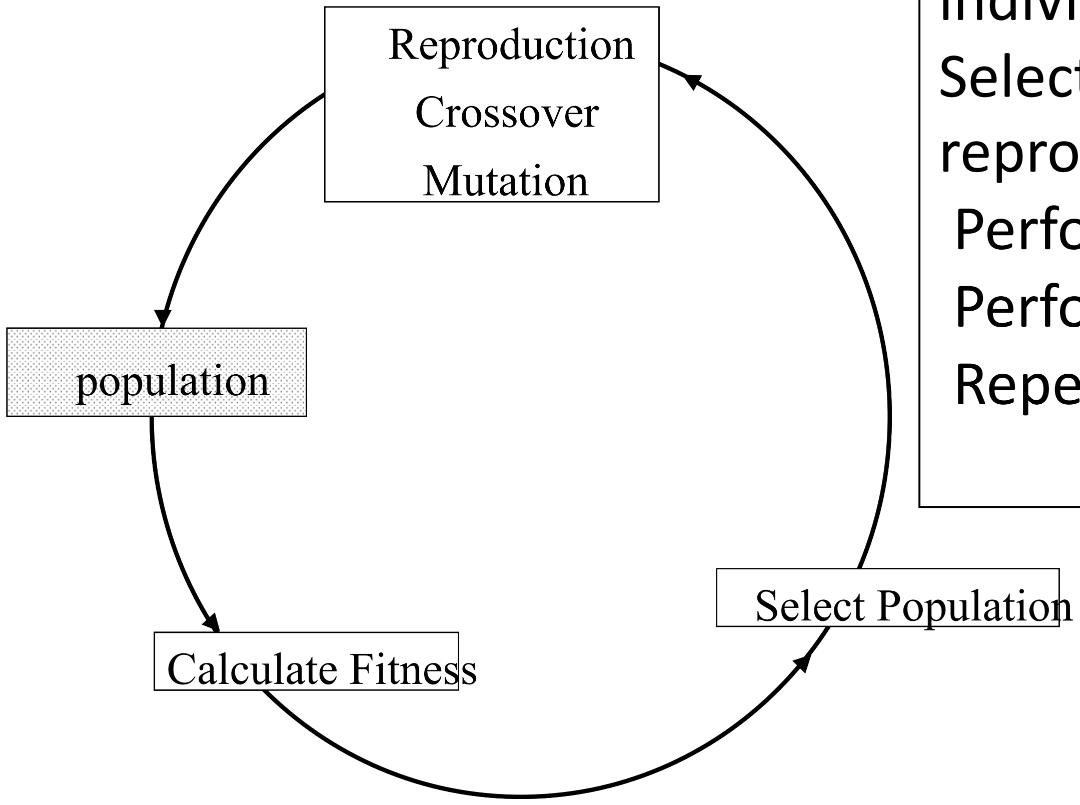
Genetic Algorithms

Inspired by Darwinian theory of evolution.

At each step of algorithm, the best solutions are selected while the weaker solutions are discarded.

Uses operators based on crossover and mutation as the basis of the algorithm to sample space of solutions.

Genetic Algorithms Steps

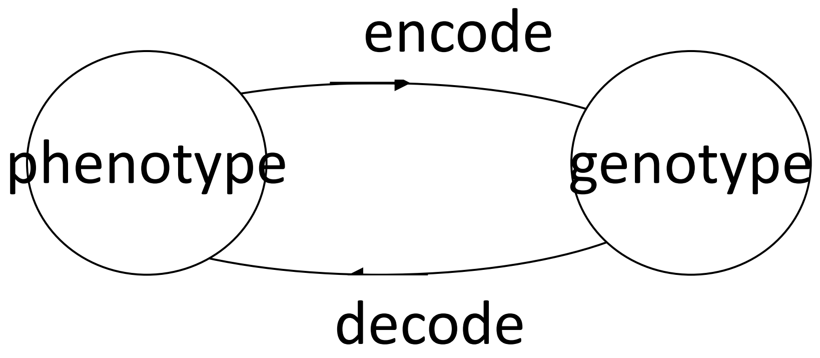


```
Create random population
while solution not found:
Calculate fitness of     each
individual
Select population for
reproduction:
 Perform crossover
 Perform mutation
 Repeat
```

Representation

Genotype

Phenotype



Traditionally, solutions are represented in binary as strings of 0s and 1s

Fitness

Need an an evaluation function which will discriminate between better and worse solutions.

Crossover

Example of one-point crossover:

11001011 and 11011***111*** gives **11001*111*** and 11011**011**

N-point:

1101101101 and *0001001000*

gives:

110*100***11***00* and *000***110***10***01**

Mutation

Occurs in the GA at a much lower rate than the crossover.

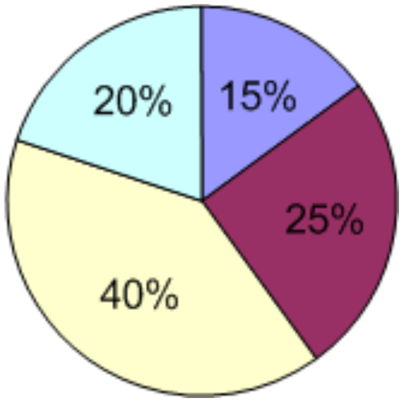
Important in order to add some diversity to the population in the hope that new better solutions are discovered and therefore it aids in the evolution of the population.

Example of mutation:

1**<u>1</u>**001001 => 1**<u>0</u>**001001

Selection

1	000110110	15%
2	111001101	25%
3	000110110	40%
4	111101111	20%



Roulette Wheel Selection:

Each sector in wheel is proportional to individual’s fitness. Selects n individuals by means of *n* roulette “turns”. Each individual is drawn independently.

Tournament Selection:

A number individuals are selected at random with replacement from the population

The individual with the best score is selected

This is repeated n times.

Issues:

- **Choice of representation for encoding individuals**
- **Definition of fitness function**
- Definition of selection scheme
- Definition of suitable genetic operators
- Setting of parameters:

size of population
number of generations
probability of crossover
probability of mutation
etc.

Genetic Programming

Genetic Programming applies the approach of the genetic algorithm to the space of possible computer programs.

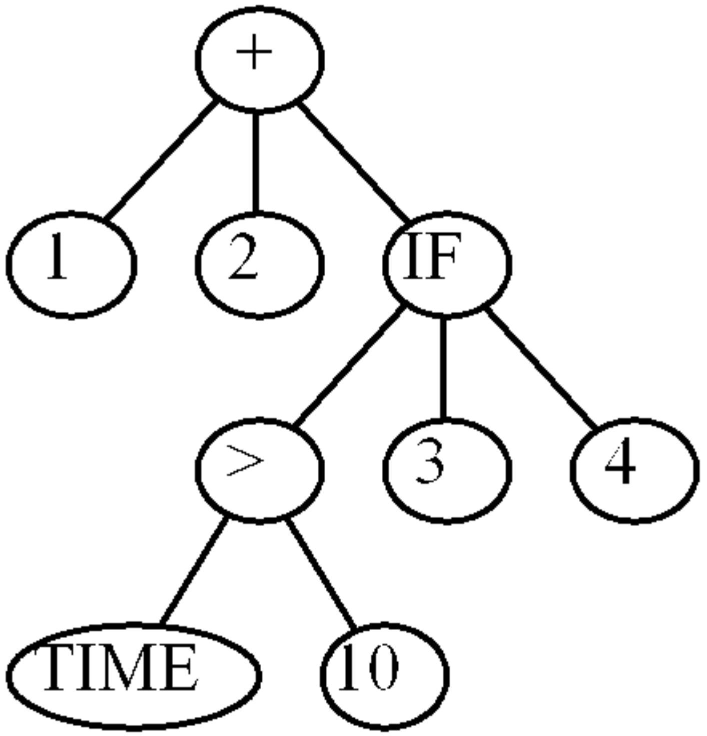
"Virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program. Genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs."

Koza

Representation

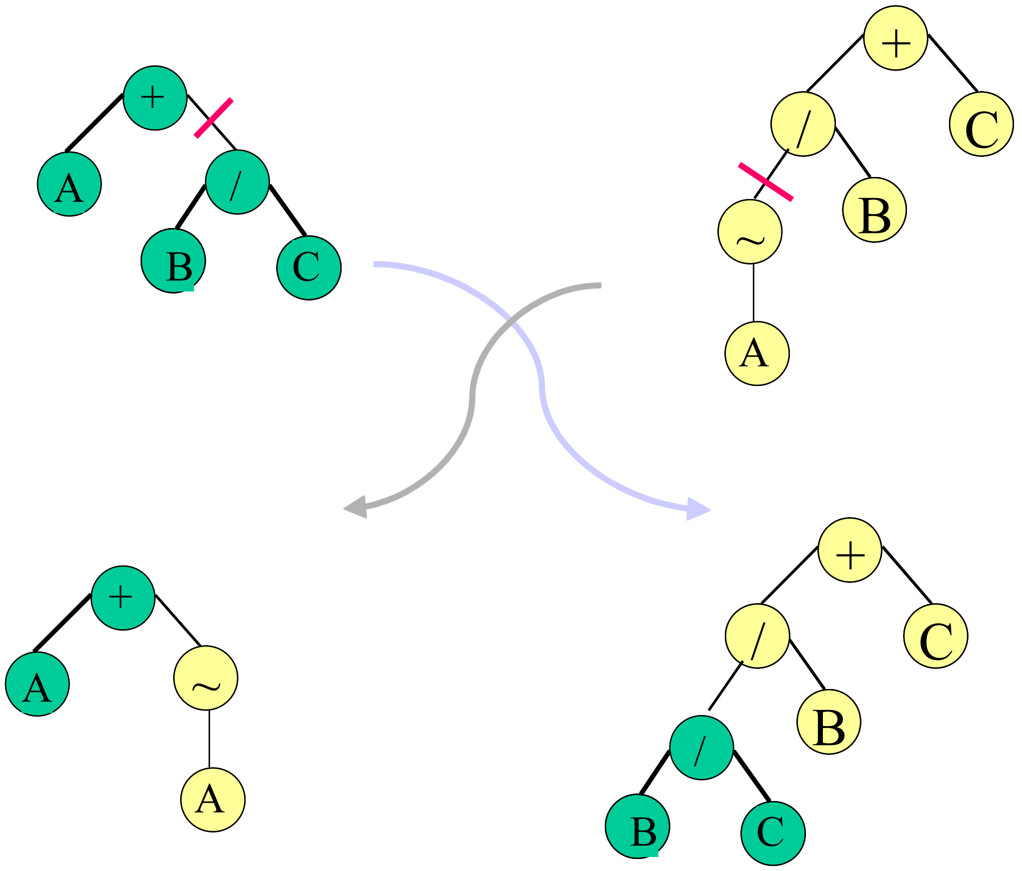
A random population of solutions is created which are modelled in a tree structure with:

- operators as internal nodes
- operands as leaf nodes

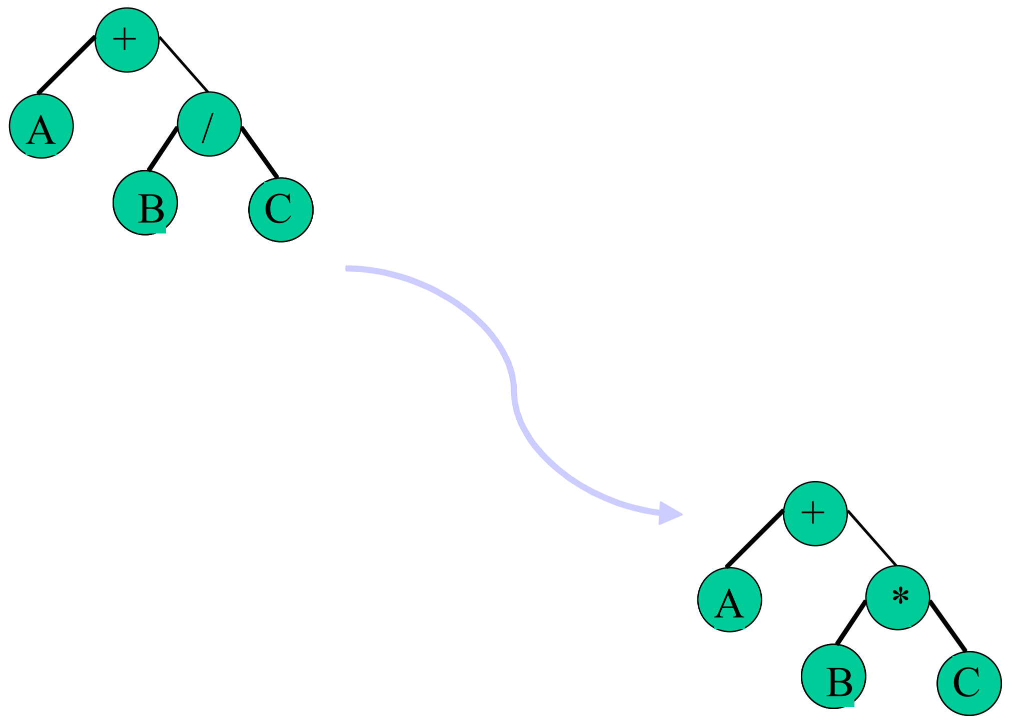


```
(+ 1 2 (IF (> TIME 10) 3 4))
```

Crossover



Mutation



Case study 1: application of genetic algorithms to IR

The effectiveness of an IR system is dependent on the quality of the weights assigned to terms in documents.

We have seen heuristic based approaches and their effectiveness and we’ve see axiomatic approaches that could be considered.

Why not learn the weights?

We have a definition of relevant and non-relevant documents; can use MAP or precision@k as fitness

Case study 1: GA approach

Each genotype can be a vector of length N (the size of the lexicon).

Set all rates randomly initially.

Run system with a set of queries to obtain fitness; select good chromosomes; crossover; mutate.

Effectively searching landscape for weights to give good ranking

Several examples: Etzioni (one of the first).

Case study 2: application of genetic programming to IR

Evolutionary computing approaches:

- evolutionary strategies
- genetic algorithms
- genetic programming

Why Genetic Programming for IR?

(Cummins, 2007)

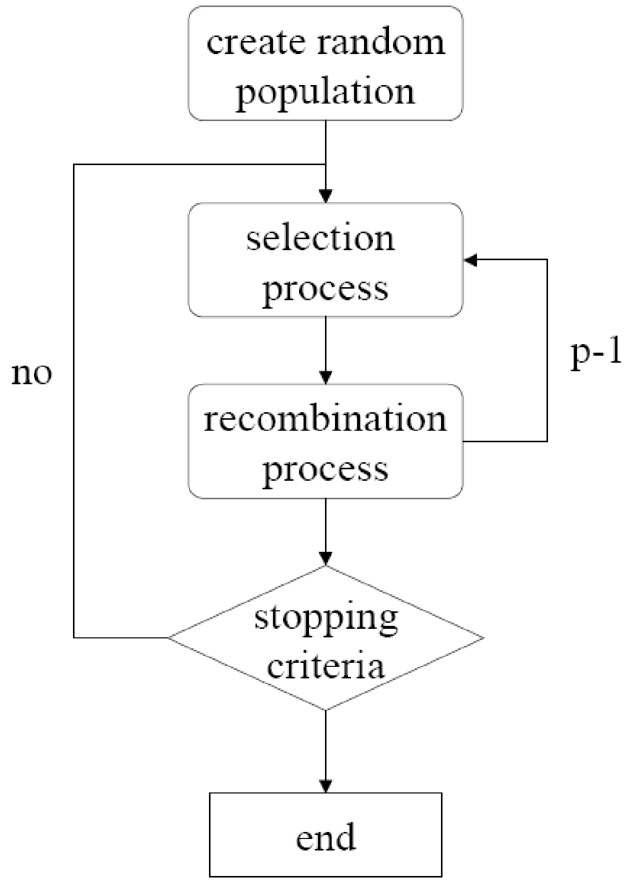
- Produces a symbolic representation of a solution which is useful for further analysis.
- Using training data, MAP can be directly optimised (i.e. used as the fitness function)
- Solutions produced are often generalisable as solution length (size) can be controlled

Genetic Programming and IR – previous approaches

- Oren (2002):
  - Non-primitive terminals, small collections, not generalisable.
- Fan *et al.* (2002-2005):
  - Mostly primitive terminals, works well for short queries only
- Trotman (2005):
  - Primitive terminals, seeded initial population, solutions more general than those previously obtained
- Almeida *et al.* (2007):
  - Non-primitive terminals, 'memorises' training data, not generalisable, ignores most of the search space, biases towards variations of already known functions, no analysis of 'evolved' solutions

Genetic Programming Flow

- Trees created at random (usually)
- Evaluate how each performs
  in its environment (using a fitness function)
- Selection occurs based on fitness
  (tournament selection)
- Crossover of selected solutions
  to create new individuals
- Repeat until population is replaced
- Repeat for *N* generations



Anatomy of a term-weighting scheme

- Typical components of term weighting schemes:

  - term frequency aspect
  - ‘inverse document’ score
  - normalisation factor

- Decompose the search space accordingly

Why separate learning into stages?

- The search space using primitive measures and functions is extremely large.
- Reducing the search space is advantageous – efficiency increased
- It eases the analysis of the solutions produced at each stage.
- Comparisons to existing benchmarks at each of these stages can be used to determine if the GP is finding novel solutions or variations on existing solutions
- It can then be identified from where any improvement in performance is coming.

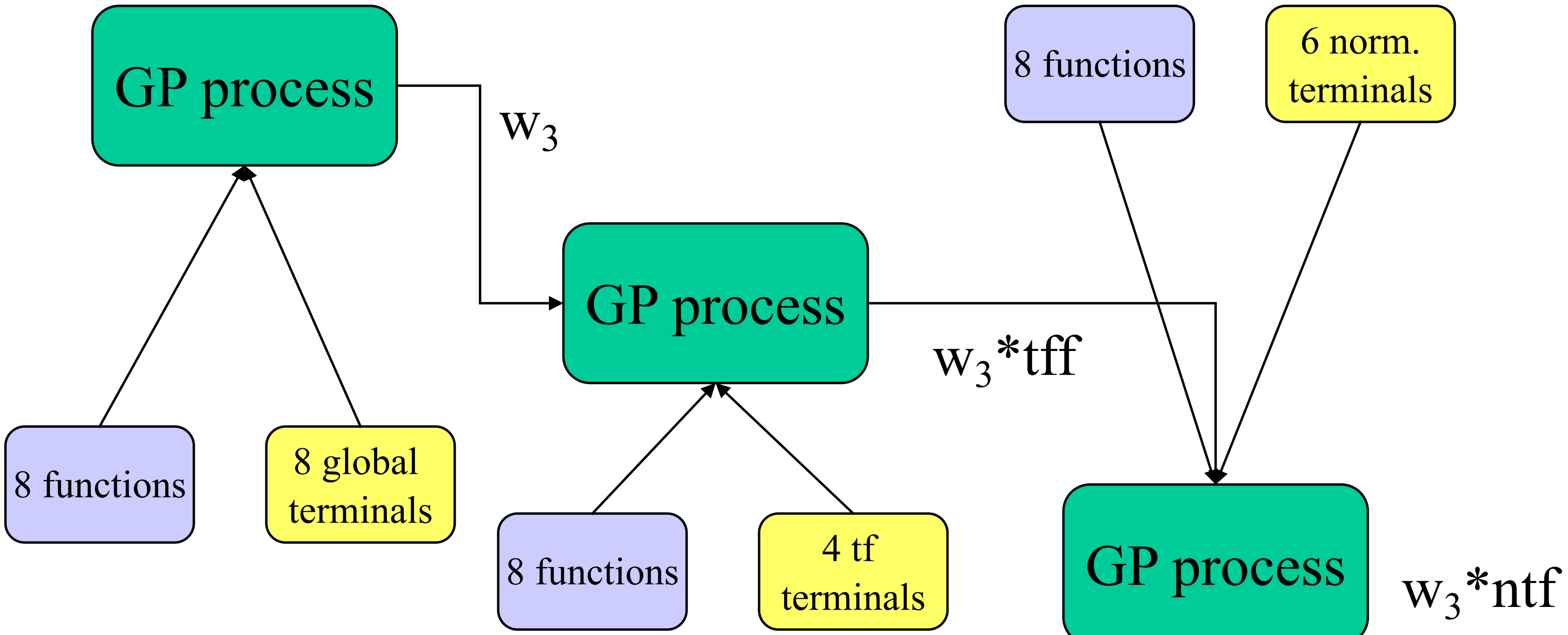
Learn each of the three parts in turn

1. Learn a term-discrimination scheme (i.e. some type of *idf*) using primitive global measures.
   1. 8 terminals and 8 functions
   2. T = {df, cf, N, V, C, 1, 10, 0.5}
   3. F = {+, *, /, -, square(), sqrt(), ln(), exp()}

2. Use this global measure and learn a term-frequency aspect.
   1. 4 terminals and 8 functions
   2. T = {tf, 1, 10, 0.5}
   3. F = {+, *, /, -, square(), sqrt(), ln(), exp()}

3. Finally, learn a normalisation scheme.
   1. 6 terminals and 8 functions
   2. T = {dl, dl_{avg}, dl_{dev}, 1, 10, 0.5}
   3. F = {+, *, /, -, square(), sqrt(), ln(), exp()}

Three-Stages



Details of Learning Approach

- 7 global functions were developed on ~32,000 OHSUMED documents
  - All validated on a larger unseen collection and the best function taken
  - Random population of 100 for 50 generations
  - Fitness function used was MAP
- 7 *tf* functions were developed ~ 32,000 LATIMES documents
  - All validated on a larger unseen collection and the best function taken
  - Random population of 200 for 25 generations
  - Fitness function used was MAP
- 7 normalisation functions were developed 3 x ~10,000 LATIMES documents
  - All validated on a larger unseen collection and the best function taken
  - Random population of 200 for 25 generations
  - Fitness function used was average MAP over the 3 collections

Analysis (1)

- Global function (w_3) always produces a positive number

$$w_3 = \sqrt{\frac{cf_t^3 \cdot N}{df_t^4}}$$

- Based on w_3 the following term-frequency factor was learned

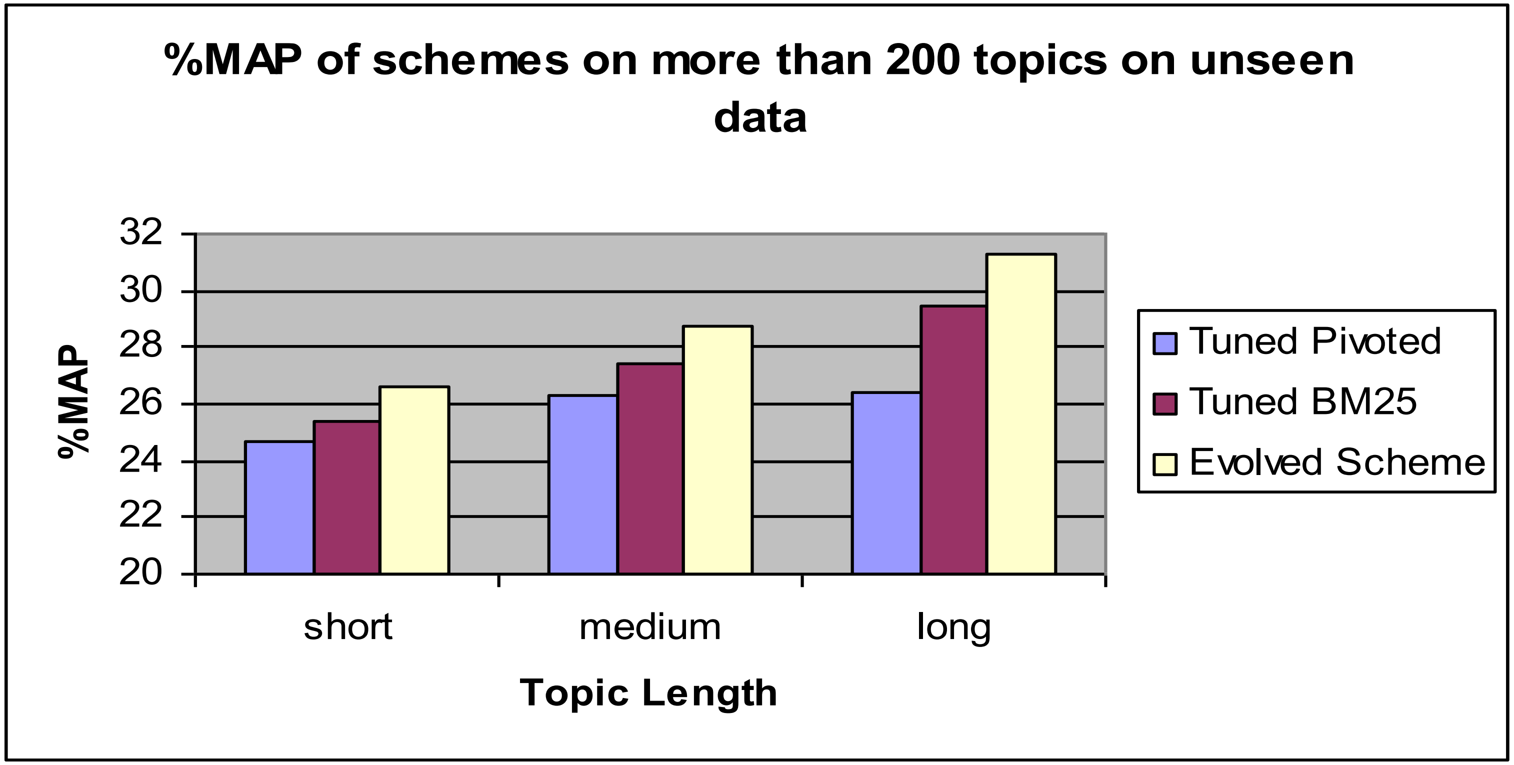
$$ntf(tf_t^D) = 10 + \frac{log(0.5)}{tf_t^D} + \frac{log(tf_t^D)}{log(1 + tf_t^D)} + \frac{log(tf_t^D)}{log(1 + tf_t^D) \cdot log(log(10))}$$

Analysis (2)

- The following normalisation scheme was found on several independent runs of the GP
- The best 6 (of the 7) schemes contained a sub-linear normalisation function shape.

$$n(dl) = \sqrt{\frac{dl}{dl_{avg}}}$$

Results on Unseen Test Data



The increase over the tuned BM25 scheme is significant ($p<0.05$) albeit small for some query lengths

Summary of GP applied to IR

Empirical evaluations shows that the evolved scheme outperforms a tuned pivoted normalisation scheme and a tuned BM25 scheme.

The evolved scheme is also nonparametric.

The use of primitive atomic measures and basic function types is crucial in allowing the shape of term-weighting functions to evolve.

Learning to rank: software

Letor

- Developed by Microsoft research asia

- Package with built in functions and terminals, data sets and learning mechanisms

Letor

- *OHSUMED, MEDLINE subset for IR*
  – 350K records from 270 medical journals from 1981-1991
  – Title, abstract, MeSH indexing terms, author, source, publication type
  – 106 queries
  – Judgments of definitely, partially, and not relevant
  – 16,140 query-document pairs with judgments
- *.GOV collection*
  – 1 million records, with 11 million links
  – 125 queries

Letor

- Very large list of built-in features (mainly non-primitive)
- Includes high level features (page rank of a page, tf-idf score, bm25 etc.)
- Currently several groups using letor
- Issue surround the correct choice of primitives

Overall summary

Many difficult problems in IR, with many
Parameters and many unknown interactions
between these parameters (e.g. term weighting,
and many others)

Learning mechanisms can be used to search this
huge space in a useful manner

Shown to arrive at solutions that outperform the
best human designed ones

Much success in classical IR, feedback, expert search etc.