Engene: A genetic algorithm classifier for content-based recommender systems that does not require continuous user feedback

John Pagonis
Pragmaticomm Limited,

September 8th, 2010
10th UK Workshop on Computational Intelligence

Engene

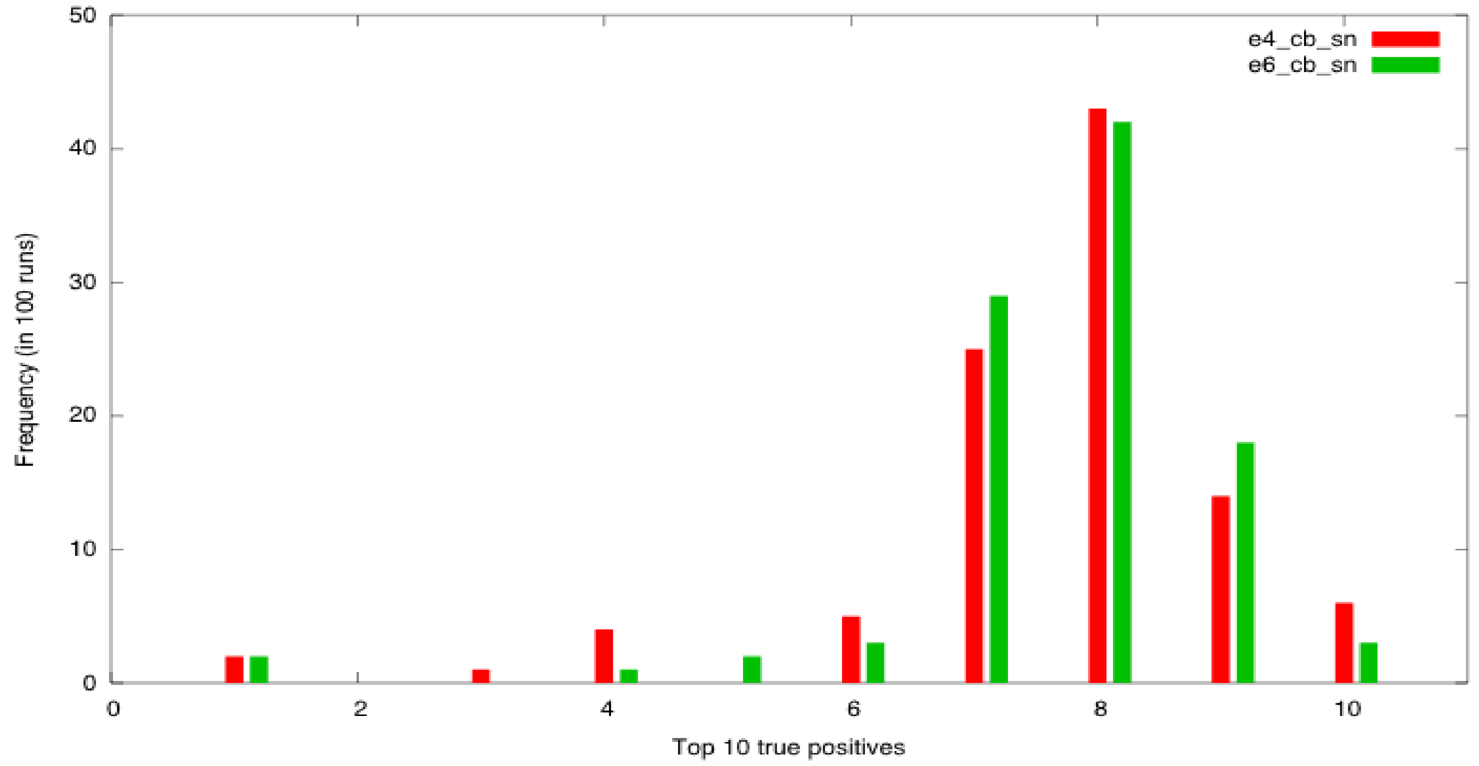
- Is a GA-based one-class soft textual document classifier that may be used either incrementally or in batch mode.
- Is primarily used incrementally as part of a personal content-based recommender system.
- Operates unattended without needing constant user feedback.

Purpose

- Engene is concerned with filtering content so that a user is presented with interesting content.
- As a one-class soft classifier Engene is applied as a filter to incoming documents and ranks them according to their pertinence to a given category.
- Ranking is achieved using the vector space model and cosine similarity function which are popular in the field of information retrieval.

Engene as a one-class batch text classifier

- Achieves a mean of 7.65 with std.dev. 1.39 in top-10 ranking tests, giving a precision of 76.5% with 17.38% recall.



- There is a 92% chance that Engene will produce between 7 and 10 true positives in the top-10 list.

Engene's performance as a batch classifier compared...

- For the same corpus used to test Engene, a variant of the One-class k-NN classifier performs better, with 90% precision at 20% recall, albeit being 5+ times slower.
- So, there is still room for improvement...
- **... but**

Engene's performance as a batch classifier compared...

- For the same corpus used to test Engene, a variant of the One-class k-NN classifier performs better, with 90% precision at 20% recall, albeit being 5+ times slower.
- So, there is still room for improvement...
- ... but we also found that Engene's False Positives are in most cases Near True Positives! This needs more investigation though.
- This is a desired property for recommender systems!.
- The whole point is to filter AND explore the search space, because predictable recommendations have little value!

Fact

- Genetic Algorithms in the past decade have been under-utilised in the fields of recommender systems and text categorisation.

Why?

Problem with using GAs in text classification

- Typically GA doc classification measures performance by asking the user to give feedback after each/few generations.

Problem with using GAs in text classification

- Typically GA doc classification measures performance by asking the user to give feedback after each/few generations.
- So the user becomes part of the fitness function
- Therefore the supervision loop is opened

… => This is not practical for “outside the lab” deployment

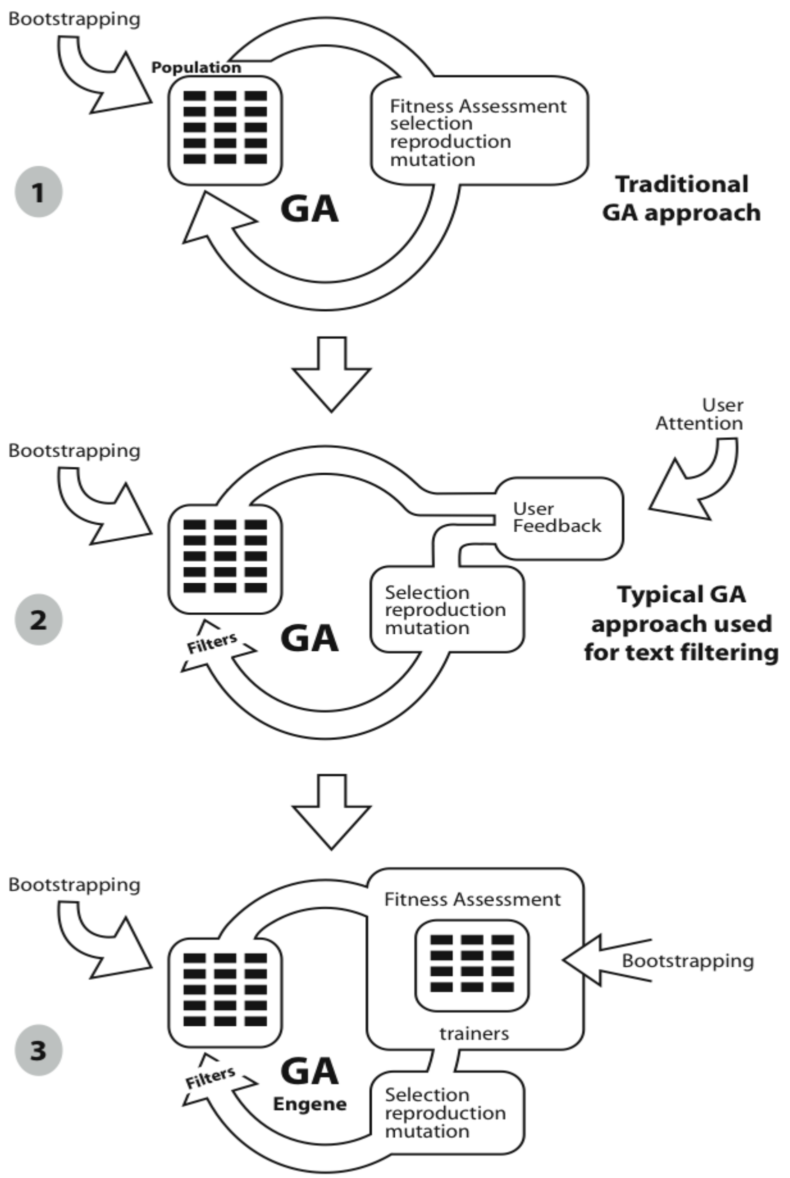
Closing the supervision loop...

- What if instead of the actual user we could use a proxy to the user?

Closing the evolution feedback loop

- Engene creates a proxy to the user (and therefore closes the supervision loop) by using a collection of documents that represent the user's feedback.
- In this arrangement 2 multi-dimensional vector sets are used to represent each user interest (class). This includes :
  - … a document collection that never evolves (in the EC sense), referred to as the trainer set
  - … a population of information filters (the trainees) which is evolved under the direction of the trainer set. These filters are genetic algorithm individuals.

GAs in text classification open the feedback loop and Engene closes it



User is removed from the fitness function

- ...**but** with removing the user from the fitness function and the need for constant attention, proper bootstrapping bcomes vital and needs much more work (if not automated)
- **...but** bootstrapping is a GA-based text classifier's dirty secret anyway...

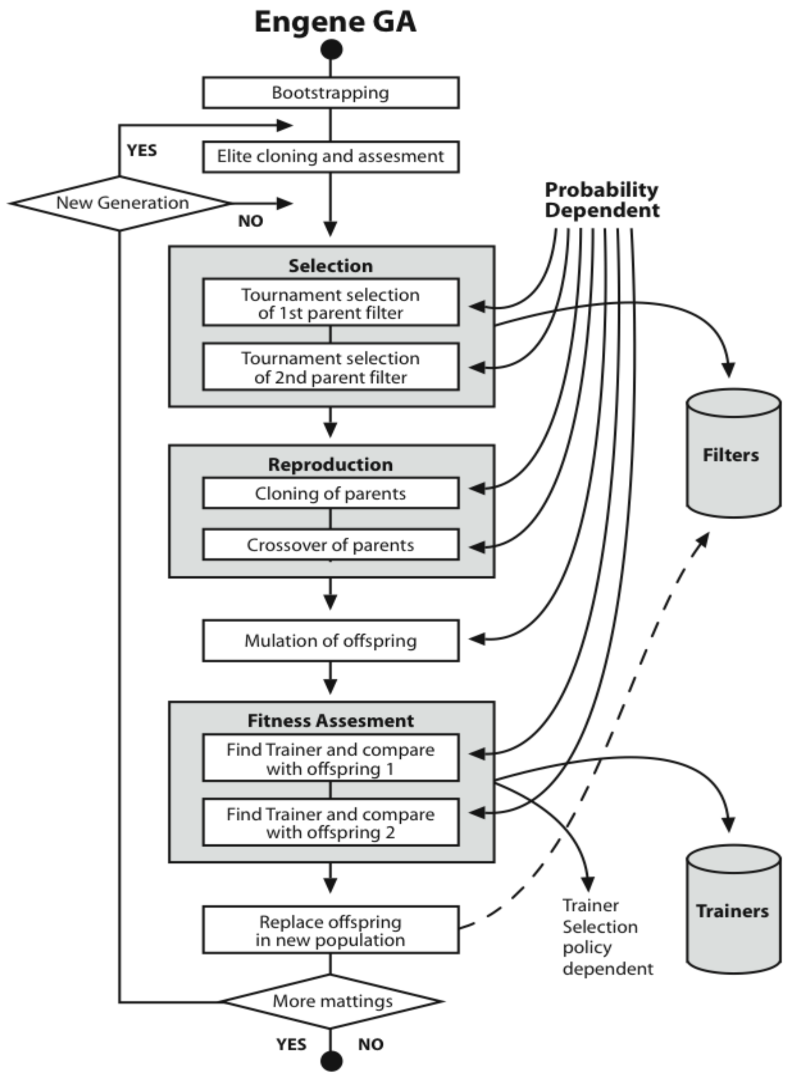
- For each subject of interest the user has to supply 20 to 30 documents that represent that interest.

- When used in batch mode, this is all that is needed to build the classifier. After 100 generations a small elite of the evolved filters is called to classify the test documents.

Engene's GA configuration

- dual population of vectors (per profile) that encode documents using the vector space model (i.e it uses multi-dimensional weighted term vectors)
- fitness function is the cosine similarity measure that assesses every filter's fitness by its similarity to a trainer
- generational by design
- variable length chromosomes
- (out of 4) tournament selection
- a random two-point crossover (at 40%)
- elitism with re-assessment needed due to xo operator (at 10-30%)
- mutation operator randomly changes a gene's allele by 20% (at 3%)
- simple cloning (at 60%)

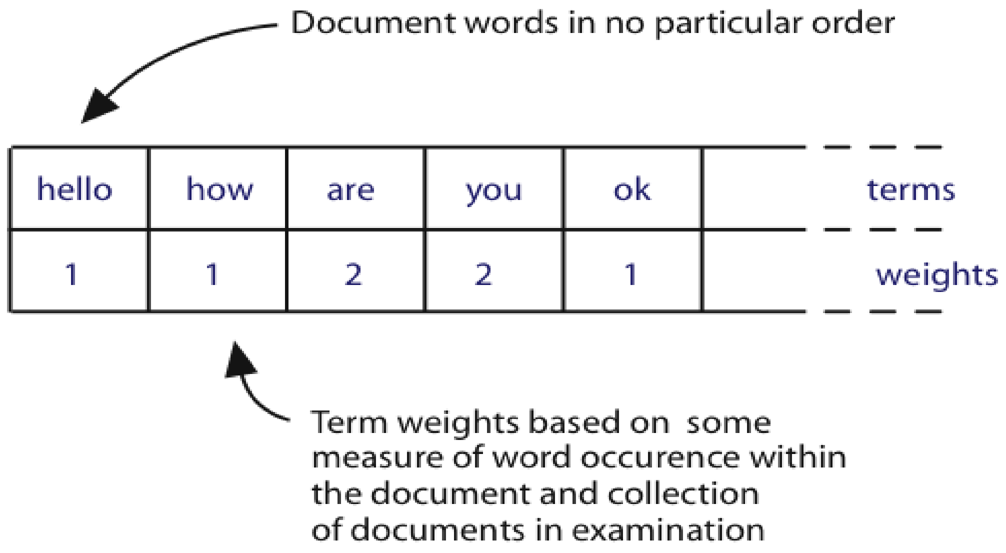
Engene GA



Engene GA
Bootstrapping
YES
Elite cloning and assesment
New Generation
NO
Probability Dependent
Selection
Tournament selection of 1st parent filter
Tournament selection of 2nd parent filter
Filters
Reproduction
Cloning of parents
Crossover of parents
Mulation of offspring
Fitness Assesment
Find Trainer and compare with offspring 1
Find Trainer and compare with offspring 2
Trainers
Replace offspring in new population
Trainer Selection policy dependent
More mattings
YES
NO

Encoding documents and individuals

- A typical way of representing documents in the vector space model is that of encoding them in weighted term vectors. In which case every document term represents a dimension of the vector while its weight denotes its scalar size.



Document words in no particular order
hello
how
are
you
ok
terms
1
1
2
2
1
weights
Term weights based on some measure of word occurence within the document and collection of documents in examination

- This fits naturally to the GA chromosome metaphor.

Term weighting

- The most popular weighting scheme for a term is that of using its Term Frequency within a document multiplied by the Inverse Document Frequency of the term in the corpus of all documents examined.
- So that:

$$W_{term} = TF*IDF$$

TF*IDF

TF : denotes how popular the word is in this document

IDF: denotes how unique (therefore discriminating) is a word in a collection of documents

most basic form of TF*IDF is:

$$W_{term} = TF * \log_{10}(N/DF)$$

where:

TF is the occurrence count of the term in the document

N is the number of documents examined

DF is the number of times that the term occurs in the document collection

... so TF*IDF is perfect for Information Retrieval

TF*IDF

TF : denotes how popular the word is in this document

IDF: denotes how unique (therefore discriminating) is a word in a collection of documents

most basic form of TF*IDF is:

$$W_{term} = TF * \log_{10}(N/DF)$$

where:

TF is the occurrence count of the term in the document

N is the number of documents examined

DF is the number of times that the term occurs in the document collection

... so TF*IDF is perfect for Information Retrieval

but...

TF*IDF vs Engene

For a small number of documents that are supplied by a user in order to define an interest (class), frequently the important terms are repeated in all documents.

thus $DF \approx N$ hence $\log_{10}(N/DF)$ --> zero

TF*IDF vs Engene

For a small number of documents that are supplied by a user in order to define an interest(class), frequently the important terms are repeated in all documents.

thus $DF \approx N$ hence $\log_{10}(N/DF)$ --> zero

Hence we discovered that using <u>no</u> IDF or $\log_{10}(N^2/TF)$ instead yields excellent results. We also normalise the TF by the term length of the vector to cater for variable length individuals.

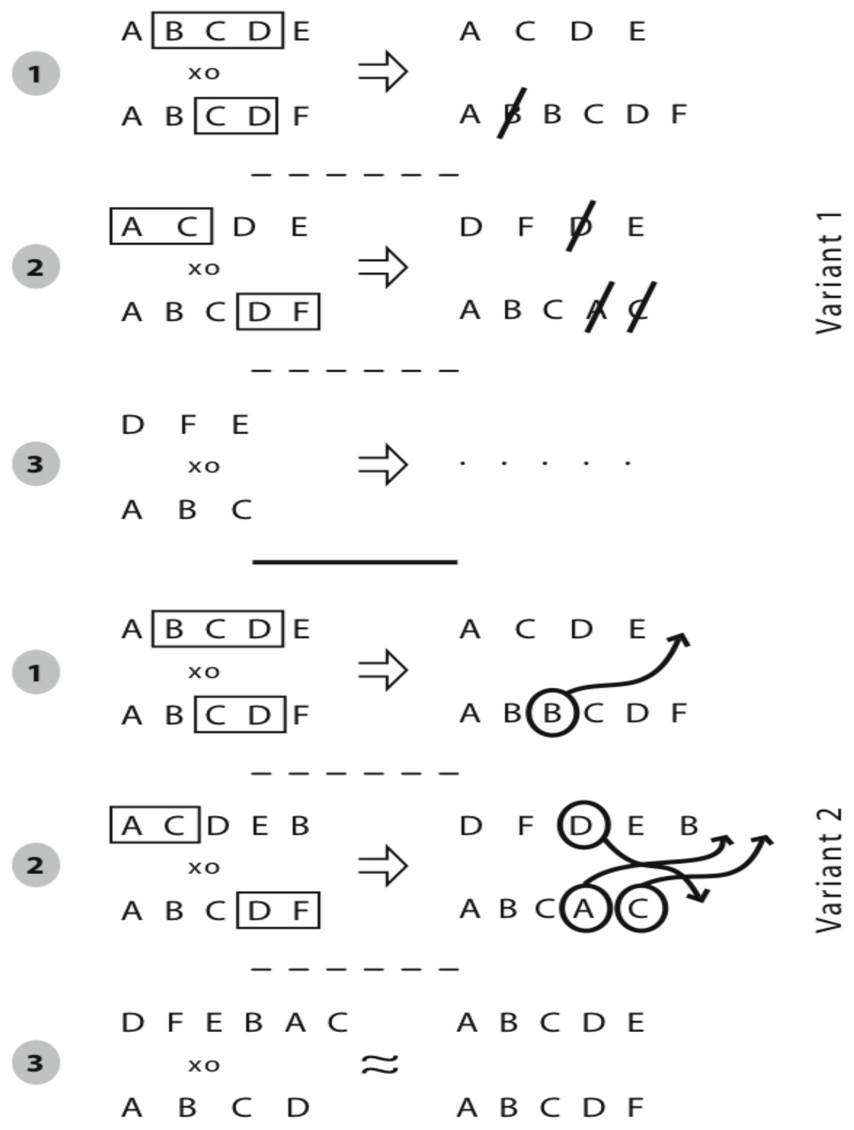
Therefore we concluded that the effects of the classic IDF are damaging during evolution, though useful during final ranking of incoming documents.

Engene's random 2-point crossover

- we tested 2 variants
- The first prunes common terms after recombination by throwing away common terms
- The second moves redundant terms back to the individual where they came from

In all cases the first performed better

Engene's random 2-point crossover



Engene's random 2-point crossover

Q: So what magic does this simple Engene GA do to create better filters?

Engene's random 2-point crossover

Q: So what magic does this simple Engene GA do to create better filters?

A: Feature Selection + Dimensionality Reduction

...but without losing any of the gene pool!

Bottom line

Engene (like some other GA based classifiers) produces shorter more focused filters, thus doesn't suffer from over-fitting.

Using the classic basic TF*IDF weighting harms this process during evolution.

These filters (vectors) tend to include the most important terms and therefore can be used with IR techniques and in combination with other classifiers to retrieve filtered sets of documents.

At the same time when feature selection is performed, information is not thrown away because it remains in the gene pool. This is desired when used in incremental mode.

Engene does so without constant attention from the user.

So why use GAs in recommenders after all?

- No over-fitting
- Serendipity
- Excellent exploration as well as good filtering
- Excellent adaptation to concept drifts when docs are added to the gene pool
- No premature convergence towards an average profile (but many populations are needed, one for every class)
- Gene pool (terms) diversity

- Thank you :-)

- www.pagonis.org/Publications.html