Finding a Needle in the Blogosphere: An Information Fusion Approach for Blog Distillation Search

José M. Chenlo[a,*], Javier Parapar[b], David E. Losada[a], José Santos[b]

[a]*Centro de Investigación en Tecnoloxías da Información (CITIUS) University of Santiago de Compostela, Rúa de Jenaro de la Fuente Domínguez, 15782 - Santiago de Compostela Spain*
[b]*Information Retrieval Lab, Department of Computer Science, University of A Coruña, Campus de Elviña, 15071 - A Coruña, Spain*

Abstract

In the blogosphere, different actors express their opinions about multiple topics. Users, companies or editors socially interact by commenting, recommending and linking blogs and posts. These social media contents are increasingly growing. As a matter of fact, the size of the blogosphere is estimated to double every six months. In this context, the problem of finding a topically relevant blog to subscribe to becomes a Big Data challenge. Moreover, combining multiple types of evidence is essential for this search task. In this paper we propose a group of textual and social-based signals, and apply different Information Fusion algorithms for a Blog Distillation Search task.

Information fusion through the combination of the different types of evidence requires optimisation for appropriately weighting each source of evidence. To this end, we analyse well-established population-based search methods. Namely, global search (Particle Swarm Optimisation and Differential Evolution) and a local search method (Line Search) that has been effective in various Information Retrieval tasks. Moreover, we propose hybrid combinations between the global search and the local search method and compare all the alternatives following a standard methodology. Efficiency is an imperative here and, therefore, we focus not only on achieving high search effectiveness but also on designing efficient solutions.

Keywords: Blogosphere, Blog Distillation, Particle Swarm Optimisation, Differential Evolution, Line Search, Hybridisation, Information Retrieval

*Corresponding author. Tel.: +34 881 816 396; Fax: +34 881 816 405.
Email addresses: josemanuel.gonzalez@usc.es (José M. Chenlo), javierparapar@udc.es (Javier Parapar), david.losada@usc.es (David E. Losada), santos@udc.es (José Santos)

1. Introduction

The blogosphere has become a key source of information to follow new trends and opinions about a wide variety of topics [1]. The raise and success of social networks (e.g., Facebook or Twitter) has motivated the evolution of blogs from informal discussion or informational sites to professional platforms specialised in specific domains (e.g., technology, fashion or finances). The proliferation of platforms to create and manage blogs, such as Blogger[1] or WordPress[2], facilitates the development of immense communities of blogs on the Internet. This explosion of specialised information sources complicates the identification of new blogs related to people's interests.

Seeking information in the blogosphere is a Big Data problem. Because of the large number of publishers and followers, because of the huge volume of contents and topics, and because of the variety of signals needed to automatically estimate relevance. For instance, a blog search engine can be tracking more than 182 million blogs[3]; and the number of blogs doubles every six months [2].

Several studies have focused on search tools for the blogosphere [3, 4]. In this area, different features and algorithms have been proposed and tested for detecting blogs potentially relevant to specific interests. For instance, some authors use the information provided by blog feeds[4] to estimate the relatedness between a blog and a particular topic [5]. Other authors go beyond these feeds and try to measure the relatedness between the topic and individual blog posts [6]. The use of a more heterogeneous set of features is also common. For instance, blog timestamps, link analysis or information provided from external resources (e.g., Wikipedia [5]). This array of experimental studies makes it difficult to understand what features are effective, and when and how they are best used. To shed light on these issues, in this paper we investigate different methods to automatically and effectively combine multiple sources of evidence to recommend blog posts based on a particular interest. We combine evidence of relevance –at blog and post level– with more heterogeneous signals such as temporal or social features.

This combination of different types of evidence entails an optimisation task for setting the most appropriate weight to each source of evidence. To face this challenge, we have selected well-established population-based methods providing a global search (Particle Swarm Optimisation and Differential Evolution); and a local search method (Line Search) that has been effectively employed in Information

[1] `https://www.blogger.com`

[2] `https://wordpress.com/`

[3] `http://smartdatacollective.com/matthewhurst/44748/farewell-blogpulse`

[4] A blog feed is a collection of the last entries –the blog's posts– from a particular blog.

Retrieval. The focus of the paper is not on providing a comparison of a wide range of search algorithms and variants of these. We have rather selected popular algorithms and standard implementations and, additionally, we implemented a hybrid version for integrating the advantages of global and local search. Besides search effectiveness, efficiency is an imperative in our application domain. Therefore, we will report on both effectiveness and efficiency results.

The main contributions of this paper are:

- We propose a set of relevance-based and social-based signals for estimating the relevance of blogs.
- We propose a fusion approach that combines these information signals for finding potentially relevant blogs to subscribe to.
- We evaluate different optimisation algorithms for weighting different features.
- We perform a thorough analysis of performance of the Information Fusion methods. Our evaluation considers both search effectiveness and efficiency.

The rest of the paper is organised as follows. Section 2 reviews some papers related to our research. In section 3, we present our fusion methods and explain the approach to combine evidence, the search models utilised, and the features proposed to estimate the relevance of a blog feed. The experimentation design and results are reported in section 4. The paper ends with section 5, where we expose some conclusions.

2. Related Work

2.1. Blog Distillation Task

The blog distillation task is defined as the process of searching for blogs with a recurring central interest, typically expressed as a textual query [3]. The task can be summarised as: *Find me a blog with a principle, recurring interest in T*. For a given target topic T, systems should suggest feeds that are devoted to T [3].

This problem can be seen as a particular instance of a resource selection problem in distributed Information Retrieval [7], in which the goal is to select the blogs that more likely contain documents relevant to the query T [1]. Many researchers utilised both the information provided by blog feeds and the information provided by the blog posts referenced by the feeds. For instance, Elsas et al. [5] combined relevance scores at feed level and at blog post level, and employed Language Models (LMs) to retrieve feeds related to a specific query. They also proposed a Query

Expansion (QE) technique based on data extracted from the Wikipedia. This technique was effective and, in fact, the best performing systems in the TREC blog distillation task applied expansion [3, 4].

In [8], Seo and Croft also followed the resource selection principle and considered different blog representations. Two main models were tested: 1) a blog represented as the concatenation of all its posts, and 2) a blog represented as a query-dependent cluster (containing only highly ranked blog posts for a given query). They found that the combination of both models was the most effective approach.

Other studies addressed the problem as an expert finding task [9]. Given a blog post and a query, the estimated relevance of the blog to the query can be seen as an indication of the interest of the post's author with respect to the query topic [1]. This approach was explored by Macdonald and Ounis [10], who defined a Voting Model [11] that represents the blogger (the blog writer) as a concatenation of all blog posts written by him. A query is run against all blog posts and every blog post retrieved that belongs to the profile is considered as a vote for the relevance of that blog. Combining evidence from feeds and posts has also been studied for other social media tasks, such as trend detection [12].

In this paper we explore an alternative approach to the blog distillation task. We consider blog distillation as an Information Fusion problem where multiple types of signals need to be combined. We study what signals are effective, and when and how they are best used to assess the recurring interest of a blog in a specific topic. We evaluate different fusion methods –with relevance-based, social-based, and temporal-based evidence– and compare their performance with that of baseline and state-of-the-art methods.

2.2. Combining evidence

A straightforward way to combine evidence in IR is linear interpolation [13]. However, optimising these models is a major bottleneck within the retrieval process. The traditional methodology in IR implies that parameter values are optimised with a training sample (for a certain evaluation measure); and, then, the trained parameter values are evaluated with a test sample. With one or two parameters, an exhaustive search for the best parameter setting –e.g., through line search– is expensive but affordable. But the computational cost of the optimisation process grows exponentially with the number of parameters. This problem has been previously addressed in the literature. Taylor et al. [14] studied retrieval models containing up to 375 parameters and agreed that the lack of efficient algorithms for parameter optimisation is one of the bottlenecks of current IR research.

The problem of combining multiple sources of evidence has been widely addressed in the literature of Information Retrieval (IR) and Machine Learning (ML). A well-known family of methods that naturally allows the combination of evidence

is Learning To Rank (L2R). L2R [15] has received much attention lately as a result of the popularity of Web search engines. These methods learn how to combine predefined features for ranking by means of discriminative learning algorithms. L2R models can be grouped into three approaches. The *pointwise* approach assumes that we want to predict the relevance degree of each individual document. Regression-based algorithms and classification-based algorithms are examples of pointwise approaches [16, 17]. The *pairwise* approach [15, 18, 19, 20, 21] takes a pair of documents and predicts their pairwise preference (e.g., +1 or -1). The *listwise* approach [22, 23, 24, 25, 26] takes the entire set of documents associated with a query and predicts the complete ranking. The listwise approach is the most natural method to rank documents in decreasing order of estimated relevance; and experimental results have showed that it has certain advantages over other algorithms [15, 26]. In this paper, we follow a listwise approach and learn a ranking function from a certain set of signals. Given a training set of blogs and a vector of features, we directly optimise a global measure of performance –Mean Average Precision (MAP)–, which is computed over the ranked set of documents as a whole.

Evolutionary algorithms [27] have been applied to learn ranking functions. For instance, Particle Swarm Optimisation (PSO) was successfully applied in IR [28, 29]. In [30], the authors experimented with a hybrid PSO-Line Search model and proposed a cooperative Line Search-Particle Swarm Optimisation (CLS-PSO) algorithm by integrating local Line Search and basic PSO. The Line Search algorithm was applied to a subset of the PSO population. The performance of the proposed hybrid algorithm –examined through four common nonlinear optimisation problems– was better than that of the standard PSO. On the other hand, Differential Evolution (DE) methods were also applied in search domains. Bollegala et al. [26] proposed a ranking method that used DE to learn a function that ranks a list of documents retrieved by a Web search engine. Their method outperformed previously proposed rank learning methods that used evolutionary computation algorithms such as PSO and Genetic Programming (GP). Another evolutionary options popularly chosen are Genetic Algorithms (GA) [31] and Fuzzy Genetic Algorithms [32]. We decided not to use GA methods because the primary focus of this paper is on efficiency. We have not experimented with Fuzzy GA either because neither we have used a classical GA for the optimisation of the weight parameters nor we used a fusion model that can be modelled as fuzzy.

In our current endeavours, we have opted to compare Line Search, Particle Swarm Optimisation and Differential Evolution, as well as a hybrid approach that combines the global search methods and Line Search.

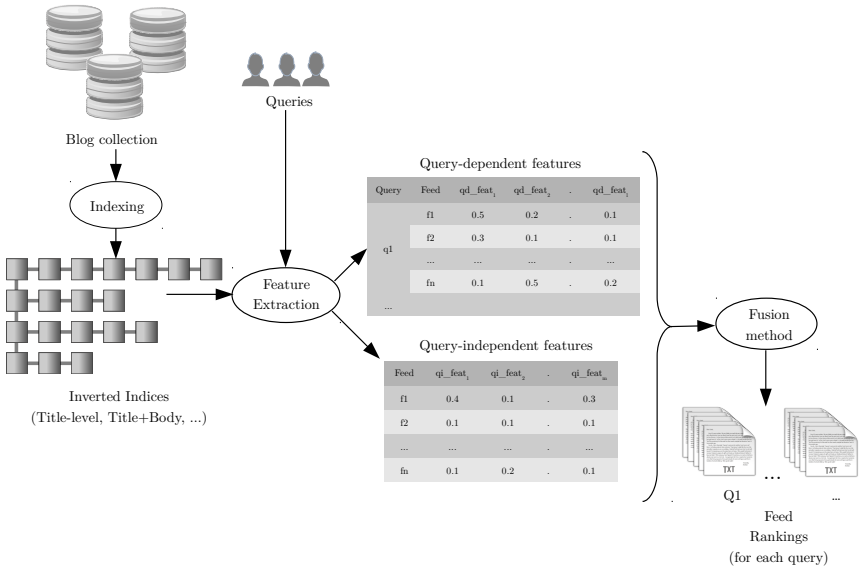


Figure 1: General framework for solving the Blog Distillation task using fusion mehtods.

3. Fusion Method

Our general framework for solving the blog distillation problem is sketched in Figure 1. First, we briefly describe the interpolation approach used to combine evidence. Second, we explain the Information Retrieval models that are required to compute some features. Third, we report the set of features proposed to estimate the relevance of a blog feed. And last, we will comment on different optimisation methods.

3.1. Combining evidence

Some of the selected features represent content-match evidence, which is dependent on the query topic. Other features, such as time-based or social-based features, are query-independent.

To effectively combine the features we need the scores in the same range ([0,1]). To meet this aim, we apply the following normalisation for query-dependent features:

$$qd\_feat_{k(norm)}(F,Q) = \frac{qd\_feat_k(F,Q)}{max_{F_i \in C}\ qd\_feat_k(F_i,Q)} \quad (1)$$

where F is a feed, Q is the query, $qd\_feat_k(F,Q)$ is one of the query-dependent features (presented next) and C is the collection of feeds.

Query-independent features are normalised as follows:

$$qi\_feat_{k(norm)}(F) = \frac{qi\_feat_k(F)}{max_{F_i \in C}\ qi\_feat_k(F_i)} \quad (2)$$

where F is a feed, $qi\_feat_k(F)$ is one of the query-independent features (presented next), and C is the collection of feeds. This type of normalisation is broadly used and has been considered as a good solution since the early beginnings of IR research [33].

We follow a simple linear combination method to combine evidence from query-dependent and query-independent features:

$$Rel(F,Q) = \sum_i \alpha_i \cdot qd\_feat_{i(norm)}(F,Q) + \sum_j \beta_j \cdot qi\_feat_{j(norm)}(F) \quad (3)$$

where F is a feed, Q is the query, qd_feat_{norm} are the normalised query-dependent features, qi_feat_{norm} are the normalised query-independent features, and α_i, β_j are free parameters.

Combining evidence by linear combination is a simple but effective approach. The review done by Croft on combination methods for Information Retrieval [13] showed that linear methods produced the best overall performance for combining multiple sources of evidence.

3.2. Information Retrieval models

Two different scoring methods, BM25 and a unigram Language Model with Dirichlet smoothing, are employed for assessing the relatedness between the query and every document. BM25 is an effective IR model derived within a Probabilistic framework[5]. We used the Lemur's implementation of the BM25 matching function[6]:

$$BM25(D,Q) = \sum_{t \in Q \cap D} w \cdot \frac{(K_1+1)\,tf_{t,D}}{K_1\left((1-b)+b \times (L_D/L_{ave})\right)+tf_{t,D}} \frac{(K_3+1)\,tf_{t,Q}}{K_3+tf_{t,Q}} \quad (4)$$

[5]We applied the well-known BM25 suggested configuration (K1 = 1.2, K3 = 7, b = 0.75), which has proved to be robust in many retrieval experiments [34].

[6]`http://www.lemurproject.org/`.

where $tf_{t,D}$ is the frequency of t in document D, $tf_{t,Q}$ is the frequency of t in query Q, L_D and L_{ave} are the length of document D and the average document length in the whole collection. And w is an inverse document frequency (idf) weight defined as:

$$w = \log\left(\frac{N - n + 0.5}{n + 0.5}\right) \tag{5}$$

where N is the total number of documents in the collection, n is number of documents that contain the term t.

The unigram Language Model with Dirichlet smoothing [35] is computed as follows[7]:

$$Dirichlet(D, Q) = \prod_{i=1}^{n} P(q_i|D) = \prod_{i=1}^{n} \frac{tf(q_i, D) + \mu \cdot P(q_i|C)}{|D| + \mu} \tag{6}$$

where D is a document, n is the number of query terms, $tf(q_i, D)$ is the raw term frequency of q_i in D, $|D|$ is the document length (term count) and μ is a parameter for adjusting the amount of smoothing applied. $P(q_i|C)$ is the probability of the term q_i occurring in the collection C (usually obtained as the maximum likelihood estimator computed from the collection).

These two measures produce evolved query-document similarity scores that incorporate advanced term weighting and document length normalisation.

3.3. Feed-level Relevance Features (Query-dependent)

Content-match evidence at feed level has been widely applied in the feed distillation task. Our proposed set of features capture evidence of matching between the query and different subparts of the feeds (Titles, Titles+Bodies, Comments); and apply different search and query expansion models. Given a topic, expressed as a textual query, the following features model the relevance of the feed:

- $RelBM25Feed\_Titles$. This is the BM25 estimation of relevance between the query and the document constructed by concatenating all titles of the feed's posts. The sequence of titles in the feed is a succinct representation of the topics discussed in the blog.
- $RelBM25Feed\_Titles + Body$, $RelDirichletFeed\_Titles + Body$. These are the BM25 and Dirichlet estimations of relevance between the query and the document constructed by concatenating all titles and body content of the feed's posts. This full-text representation of the feeds considers all the content of every blog post but it is potentially noisy.

[7] We used default smoothing values: $\mu = 1000$.

- *RelBM25Feed_Comments*, *RelDirichletFeed_Comments*. These are the BM25 and Dirichlet estimations of relevance between the query and the document constructed by concatenating all comments associated to any of the blog's posts. The importance of the comments has been demonstrated in many experiments with blog datasets [36, 37].

- *ExpansionFeed_Titles + Body*. Applying Pseudo Relevance Feeedback (PRF) and Query Expansion (QE) is an efficient, effective and natural way to improve the quality of the original query [38]. In this paper we use a popular expansion method based on Relevance Models (RM1 PRF) [39]. After doing some assumptions the RM1 method is defined as:

$$P(w|R) \propto \sum_{d \in C} P(d) \cdot P(w|d) \cdot \prod_{i=1}^{n} P(q_i|d) \tag{7}$$

  $P(d)$ is the document prior (usually assumed to be uniform), $\prod_{i=1}^{n} P(q_i|d)$ is the query likelihood given the document model, which is traditionally computed using Dirichlet smoothing. $P(w|d)$ accounts for the importance of the word w within the document d. C is the collection of documents.

  The terms with highest estimated probability under the Relevance Model (R) are selected to conform the expanded query[8].

  The feature *ExpansionFeed_Titles + Body* represents the score of relevance of the feed obtained with the expanded query. The feed is represented as the concatenation of all post titles and post bodies.

- *ExpansionFeed_Comments*. This is equivalent to *ExpansionFeed_Titles+ Body* but the feed is represented as the concatenation of all comments.

3.4. Intra-feed Relevance Features (query dependent)

Rather than aggregating the contents of all blog posts, these features try to infer the relevance of a blog by analysing individual scores of relevance between the query and the blog's posts. All these features represent a blog post as the concatenation of the title and body of the post.

- $MaxRelBM25_P$. The maximum BM25 score of relevance between the query and the blog posts. The most relevant blog post is therefore the one that determines the $MaxRelBM25_P$ value of the feed.

[8] We took the top 100 terms as expansion terms and the Relevance Model was estimated from the top 100 documents retrieved given the initial query.

- $MeanRel_P$. The average BM25 score of relevance between the query and the blog posts. Rather than representing the feed with the most relevant post, this feature opts for computing the mean estimated relevance across all blog posts. In this way, we can estimate whether or not the feed's recurrent interests overlap with the query topic.

- $VarRel_P$. The variance of BM25 scores of relevance between the query and the blog posts. By modelling the matching tendency throughout the whole feed we expect to model scoring trends that distinguish relevant feeds from non-relevant ones. The variance of matching scores –e.g., query-sentence scores– has been successfully exploited in the literature of IR [40, 41].

- $\#PostsRanked_F$. This feature is based on indexing all blog posts from all feeds, running the query against this index, and counting the number of blog posts that belong to a specific feed (in the top 1000 retrieved posts). Every retrieved blog post from the feed is therefore a vote for the relatedness between the query and the feed.

This set of features helps to capture different aspects of relevance of the feeds. $MaxRelBM25_P$ and $\#PostsRanked_F$ are focused on the most salient blog posts from the feed, while $MeanRel_P$ and $VarRel_P$ analyse the whole distribution of relevance scores to determine whether the blog has a recurring interest in the query topic.

3.5. Social and Temporal Features

The flow of comments in the feed is as an important indication of the social impact of the blog posts. Popular and authoritative blogs attract attention and their posts have many comments from other users. Comments are therefore a meaningful source of evidence. As a matter of fact, blog comments have supported a wide range of tasks, such as opinion retrieval [37] or summarisation [42].

Time is another important dimension. For instance, if the most relevant blog post is too old then it is likely useless for the user. Temporal information is available in the feeds and it helps to understand what are the main topics of the blogs through time.

This leads to the following set of social and temporal features:

- $MeanComm_F$ The average of number of comments to the blog posts of the feed.

- $RelBM25Recent_P$. The BM25 score of relevance of the most recent post of the feed. This helps to determine whether or not the query topic was addressed in the newest contents from the feed.

- $RelBM25Old_P$. The BM25 score of the oldest post of the feed. In this way, we know whether or not the query topic was addressed in the early days of the feed.

- $RangeRel_P$. This feature represents the time range between the newest relevant blog post and the oldest relevant blog post. We index all blog posts from all feeds, run the query against this index, extract the newest and oldest blog post from each feed, and compute the difference (in days) between the dates of these two posts. A feed with a narrow time range only addressed the query topic during a small temporal window. This feed is probably less relevant than a feed whose time range is larger.

$MeanComm_F$ is query-independent and the other three features are query-dependent.

3.6. Optimisation Methods

Our Information Fusion approach is not parameter-free. All signals that are combined need to be properly weighted (parameters α_i and β_j in eq. 3). To optimise these parameters, we follow a methodology that is independent of the analytical form of the retrieval methods.

We select two population-based search methods, Differential Evolution (DE) and Particle Swarm Optimisation (PSO), as representative and current search methods with proved robustness and efficiency against other classical evolutionary algorithms. Since the focus of the paper is not on providing a comparison of different search algorithms and variants of these, we selected the standard DE and PSO to perform the required optimisation of the weight parameters involved in the fusion of evidences. For the same task, we also select a local search method (Line Search) useful in Information retrieval and, in addition, we implement a hybrid version for integrating the advantages of the two methods (global and local search methods), comparing the efficiency of the different alternatives.

3.6.1. Line Search

Line Search (LS) is a general class of optimisation methods that first find a descent direction along which the objective function will be reduced and then compute a step size that decides how far the parameter values should move along that direction [43]. Here, we employ the Line Search procedure proposed by Taylor et al. [14], who concluded that Line Search and Gradient Descent (which takes advantage of the knowledge about the analytical form of the retrieval function) perform similarly in terms of effectiveness.

From an initial random point in the parameter space, a search in each dimension is performed, moving each time the parameter value in one dimension while fixing

the values of the other dimensions. The procedure is summarised in the pseudo-code of Algorithm 1. For each dimension, N sample points are selected with equal inter-distance in the axis and around the initial parameter value (taking into account the limits of the parameter space). To assess the optimality of each point, a *fitness* value is calculated for each of these sample points, and the point with the best fitness is stored. The fitness function will be dependent on the problem and will measure the quality of a parameter setting for the given task. This basic procedure is repeated for each of the dimensions or parameters (Step 1 of the algorithm).

Algorithm 1 Line Search Algorithm.

N=Number of sample points in each dimension, D=number of dimensions, I=Sampling interval.
Select an initial random point.
Step 1 of the algorithm (for dimension d)
$min \leftarrow max(0, initial\_position[d] - \frac{I}{2})$; (0 *is the parameter lower limit*)
$max \leftarrow min(1, initial\_position[d] + \frac{I}{2})$; (1 *is the parameter upper limit*)
$increment \leftarrow \frac{(max-min)}{N}$;
$best\_position[d] \leftarrow min$; (*Best initial position*)
for $n \leftarrow 1$ ***to*** N **do**
    $p \leftarrow min + increment * n$;
    $new\_position[d] \leftarrow p$;
    if $(fitness(new\_position[d]) < fitness(best\_position[d]))$ **then**
        $best\_position[d] \leftarrow new\_position[d]$;
    end if
end for
return best_position[d]
Step 2 of the algorithm
for $d \leftarrow 1$ **to** D **do**
    $max\_dim[d] \leftarrow max(initial\_point[d], best\_position[d])$;
    $min\_dim[d] \leftarrow min(initial\_point[d], best\_position[d])$;
    $increment[d] \leftarrow \frac{(max\_dim[d]-min\_dim[d])}{N}$;
end for
$best\_position \leftarrow initial\_position$
for $n \leftarrow 1$ ***to*** N **do**
    for $d \leftarrow 1$ ***to*** D **do**
        $new\_position[d] \leftarrow min\_dim[d] + increment[d] * n$;
    end for
    if $(fitness(new\_position) < fitness(best\_position))$ **then**
        $best\_position \leftarrow new\_position$
    end if
end for
return *best_position*

In a second step of the algorithm, a line between the original point and the new computed point is defined. This new point is built by taking for each dimension the parameter value with the best fitness in the first step. This line represents the *promising* direction. Again, the same procedure is repeated, selecting a number of equidistant points (samples) between the two extremes of the promising segment. The point with the best fitness is selected and acts as the new starting point in the next iteration of the algorithm.

An *iteration* is defined as one cycle through all the parameters, plus the final search along the promising direction. With P parameters, the procedure performs $P + 1$ line search operations per iteration. Finally, the scale over which the samples are taken is reduced by a factor of 0.85 at the beginning of each new iteration, allowing higher exploration in the first iterations and higher exploitation in the final iterations.

3.6.2. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) [44] is a class of swarm intelligence techniques [45] inspired by the social behaviour of bird flocking or fish schooling. It is a population-based stochastic method where the potential solutions, called *particles*, fly through the problem space following the current optimum particles. The movements of the particles are guided by the best known position of each particle in the search space as well as the entire swarm's best known position. The process is repeated until a satisfactory solution is discovered.

The basic PSO algorithm is summarised in Algorithm 2. Each particle i stores its current position x_i^t, velocity v_i^t and its best known position pb_i^t at time t. The algorithm stores the best known position of the entire swarm (gb^t).

Algorithm 2 PSO basic Algorithm.

Initialise all particles i with random positions x_i^0 in search space as well as random velocities v_i^0.
Initialise the particle's best known position (pb_i^0) to its initial position.
Calculate the initial swarm's best known position gb^0.
repeat
  for all Particle i in the swarm **do**
    Pick random numbers: $r_p, r_g \in (0, 1)$
    Update the particle's velocity: $v_i^{t+1} = a * v_i^t + b * r_p * (pb_i^t - x_i^t) + c * r_g * (gb^t - x_i^t)$
    Compute the particle's new position: $x_i^{t+1} = x_i^t + v_i^{t+1}$
    if $fitness(x_i^{t+1}) < fitness(pb_i^t)$ **then**
      Update the particle's best known position: $pb_i^{t+1} = x_i^{t+1}$
    end if
    if $fitness(pb_i^{t+1}) < fitness(gb^t)$ **then**
      Update the swarm's best known position: $gb^{t+1} = pb_i^{t+1}$
    end if
  end for
until termination criterion is met
return The best known position: gb.

In the algorithm, a, b and c are constants that separately control the importance of the three directions that determine the next velocity and position of the particle. The three components are usually referred as *inertia* (v_i^t), *personal influence* ($pb_i^t - x_i^t$) and *social influence* ($gb^t - x_i^t$). By updating the velocities with some element of randomness, the exploration of novel areas of the search space is enabled. This avoids stagnation in local minima and it is achieved by injecting random values, in

the range (0,1), for the terms r_p and r_g. This yields a region of uncertainty around both positions, pb_i^t and gb^t.

PSO is an attractive solution for many optimisation or search problems. It has fewer parameters than popular simulated evolution methods, such as classic genetic algorithms [46]. Furthermore, with standard genetic algorithms, it is difficult to control the balance between exploration and exploitation. Even with low selective pressures there is a high probability that the population converges to a local optimum in few generations. Therefore, there is no guaranty that different potentially good areas in the space are simultaneously searched. Another alternative to deal with this problem comes from the niche and speciation techniques based on *fitness sharing* [46], or the distribution of the population in races or islands that evolve independently and simultaneously, which periodically interchange their best genetic material. PSO, in contrast, intrinsically explores the search space with a concentration of the population around the promising areas.

3.6.3. Differential Evolution

Differential Evolution (DE) [47][48] is a population-based search method. DE creates new candidate solutions by combining existing ones according to a simple formula of vector crossover and mutation. Then the algorithm keeps whichever candidate solution that has the best score or fitness on the optimisation problem at hand. The central idea of the algorithm is the use of difference vectors for generating perturbations in a population of vectors. This algorithm is specially suited for optimisation problems where possible solutions are defined by a real-valued vector. The basic DE algorithm is sketched in the pseudo-code of Algorithm 3.

Algorithm 3 Differential Evolution Algorithm.

Initialize the population randomly
repeat
  for all individual x in the population **do**
    Let $x_1, x_2, x_3 \in$ population, randomly obtained $\{x_1, x_2, x_3, x$ different from each other.$\}$
    Let $R \in \{1, ..., n\}$, randomly obtained {n is the length of the chain.}
    for $i = 1$ *to* n **do**
      Pick $r_i \in U(0, 1)$ uniformly from the open range (0,1).
      if $(i = R) \vee (r_i < CR)$ **then**
        $y_i \leftarrow x_{1i} + F(x_{2i} - x_{3i})$
      else
        $y_i = x_i$
      end if
    end for$\{y = [y_1, y_2...y_n]$ is a new generated candidate individual$\}$
    if $f(y) < f(x)$ **then**
      Replace individual x by y
    end if
  end for
until termination criterion is met
return $z \in$ population $\setminus \forall t \in$ population , $f(z) <= f(t)$

Differential Evolution, like PSO, requires a reduced number of parameters. The parameters are F (differential weight) and CR (crossover probability). The weight factor F (usually in $[0, 2]$) is applied over the vector resulting from the difference between pairs of vectors (x_2 and x_3). CR is the probability of crossing over a given vector of the population (x) and a candidate vector (y) created from the weighted difference of the two vectors ($x_1 + F(x_2 - x_3)$). The index R guarantees that at least one of the parameters (genes) changes when generating the candidate solution.

Unlike classical evolutionary algorithms, DE reduces parameter tuning and provides an automatic balance in search. As Feoktistov [49] points out, the fundamental idea of the algorithm is to adapt the step length ($F(x_2 - x_3)$) intrinsically along the evolutionary process. At the beginning, the step length is large because individuals are far away from each other. As the evolution goes on, the population converges and the step length becomes smaller and smaller. This provides an automatic balance in the search process.

The usual implementation of DE chooses the base vector x_1 randomly (variant $DE/rand/1/bin$), or uses the individual with the best fitness found so far (x_{best}) (variant $DE/best/1/bin$). To avoid the high selective pressure of the latter approach, we followed a usual strategy that interchanges the two possibilities across generations: $DE/rand/1/bin$ is applied when the quality of the best individual does not improve over selecting the best individual as base vector ($DE/best/1/bin$) after a given number of generations.

DE offers fast convergence, robustness, conceptual simplicity, few parameters with an easy implementation, and has a reliable control in the balance between exploration and exploitation [49].

3.6.4. Hybrid Solutions

We combined the global search of the population-based algorithms (PSO and DE) and LS to leverage the advantages of both approaches. The global search explores simultaneously in different areas of the search landscape, while LS explores in a restricted area centered on the best found individual. A natural combination is to apply Line Search to navigate in the immediate neighbourhood of the individuals of the population, performing a fine-tune of the individuals. In this way, LS acts as an exploitation operator.

Efficiency is a major concern in our large-scale problem. Therefore, we only applied LS to the best individual of the genetic population. This reduces the additional evaluations required by LS. This restricted search with the best individual is performed at the end of each generation of the evolutionary algorithm. LS's parameter N is set to a small value (4 in the experiments reported later). Thus, for each dimension, only a few additional sample points are selected around the initial

parameter value of the best individual[9]. Finally, the number of steps of LS was set to 1. This selects the best individual from the final sample points of the first promising direction (Algorithm 1, Section 3.6.1).

This hybridisation follows a Lamarckian strategy. The Lamarckian strategy means that the changes provided by an additional search (local search procedure or any heuristic) over any genotype of the population revert to the original genotypes [50]. However, since we used the combination only with the best individual, this reduces the problem of the decrease of the genetic diversity in the genetic population, which is inherent to the Lamarck strategy [50].

4. Experiments

To test our Information Fusion models we worked with benchmarks from the TREC 2007 and the TREC 2008 Feed Distillation task [3, 4]. The blog distillation task consists of finding blogs with a recurrent and specific interest in a given topic T. The task was defined as a classical IR problem in which the systems have to retrieve a ranking of 100 blog feeds related to a query (from the BLOGS06 research collection). The testbeds are composed by 45 and 50 queries, respectively. The BLOGS06 research collection [51] is a large data set (more than 140 GB) composed of blog home pages, XML feed documents and its blog entry pages (permalinks). Some statistics about this collection can be found in Table 1.

This collection is a standard benchmark to perform experiments on blog search. Each TREC topic contains three fields (title, description and narrative). We experimented with short queries obtained from the title field. These queries are good representatives of real user web queries [3, 4]. Text pre-processing was merely based on removing common words (from Fox's list of 733 stopwords [52]). We did not apply stemming.

4.1. Setup of Optimisation Algorithms

For LS we used 8 samples per dimension (parameter N) and 50 iterations or epochs. The sampling interval was decreased after each iteration by a factor of 0.85 (Algorithm 1).

For PSO we employed the recommended settings for the parameters a, b and c [53] (0.7, 0.9 and 0.9, respectively).

For DE we set CR to 0.9 and F to 0.5. These values belong to the intervals suggested by Storn and Price [48] ($F \in [0.5, 1.0]$, $CR \in [0.8, 1.0]$). The base

[9]Using a low sampling interval with value 0.5 reduced computations by a factor of 0.85 in every generation of the evolutionary algorithm.

Number of Unique Blogs	100649
RSS	62%
Atom	38%
First Feed Crawl	06/12/2005
Last Feed Crawl	21/02/2006
Number of Feeds Fetches	753681
Number of Permalinks	3215171
Number of Homepages	324880
Total Uncompressed Size	148GB
Feeds (Uncompressed)	38.6GB
Permalinks (Uncompressed)	88.8GB
Homepages (Uncompressed)	2.8GB

Table 1: Main statistics of the BLOGS06 collection. This collection was used in the TREC 2006, TREC 2007 and TREC 2008 blog tracks.

vector (x_1 in Algorithm 3) was chosen by interchanging the two main variants of DE ($DE/rand/1/bin$ and $DE/best/1/bin$). At the beginning, the variant that chooses the base vector from the best in the population is used ($DE/best/1/bin$). If the fitness of the best individual does not improve after 3 generations then DE selects a random individual from the population until an improvement is obtained (variant $DE/rand/1/bin$).

In both population-based search methods, DE and PSO, we used 128 particles and 50 generations. LS needs $P + 1$ line search operations per iteration (Section 3.6.1), P equals 15 (number of parameters to optimise, eq. 3), and we used $N = 8$. This means that 128 fitness calculations per iteration are required. So, an iteration of LS requires the same number of fitness evaluations than a generation of PSO or DE.

As indicated in Section 3.6.4, for the hybrid algorithms, we used $N = 2$ (2 samples per dimension) and 1 iteration or epoch. LS is applied only to the best individual of the genetic population; and for this best individual we need an extra $(15 + 1) \times 2$ evaluations. We used a population of 96 individuals or particles in order to have the same number of evaluations per generation.

To alleviate computational costs, we extensively exploited parallelism. The fitness scores –associated to every evolution point of the algorithms– were computed in parallel with a multi-threading approach and taking in account the necessary synchronisation point inherent to each algorithm (with the population-based methods, the evaluation of the population members was done in parallel; with LS, the evaluation of dimensions and samples was done in parallel).

4.2. Fitness Function and Metrics

We chose Mean Average Precision (MAP), a popular IR performance measure, as our fitness function. MAP is a single-figure measure that assesses the effectiveness of a given ranking of objects.

For a single information need, Average Precision is the average of the precision values obtained for the set of top k objects existing after each relevant object is retrieved. This value is then averaged over queries [54], i.e.:

$$MAP = \frac{1}{|Q|} \cdot \sum_{j=1}^{|Q|} \frac{1}{m_j} \cdot \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{8}$$

where, given the set of relevant objects for a query $q_i \in Q$, R_{jk} is the set of ranked retrieval results from the top result until you get to object o_k , m_j is the number of relevant objects for query q_j , and

$$Precision(R_{jk}) = \begin{cases} \frac{\#(\text{relevant objects retrieved in } R_{jk})}{R_{jk}} & , \text{when } o_k \text{ is relevant} \\ 0 & , \text{otherwise} \end{cases} \tag{9}$$

MAP is a recall-oriented measure, which accounts for the proportion of relevant objects that a system retrieves.

Another important aspect of search systems is their precision. Precision-oriented measures are concerned with the fraction of retrieved instances that are relevant. P@10, a popular precision measure, is the proportion of the top 10 retrieved objects that are relevant, i.e.:

$$P@10 = \frac{\#(\text{relevant objects retrieved in the top 10})}{10} \tag{10}$$

Given a set of queries, their respective P@10 values are averaged out to get a single P@10 figure.

We did not use P@10 as a fitness function but we report the P@10 performance of the different methods tested.

4.3. Results

The following experiments were run with a Supermicro Server with 4 AMD Opteron 6376 processors (64 cores in total), 512GB of DDR3 RAM and a SSD disk. We split each query set (2007 and 2008) into two equal-sized subsets, applied cross-validation, and report the performance obtained with each of the two folds. We decided to do so because applying 10-fold cross-validation would lead to tiny

	Train	Test
2007a	974-995	951-973
2007b	951-973	974-995
2008a	1076-1100	1051-1075
2008b	1051-1075	1076-1100

Table 2: Training and test configurations. Each row reports the query range included in each fold.

	2007a				2007b			
	MAP		P@10		MAP		P@10	
$BM25_{feed}$	.2215		.4304		.2266		.3409	
$\#PostsRanked_F$	.2034		.4391		.1987		.3318	
LS	.3267	(47%)▲	.6000	(39%)▲	.3732	(65%)▲	.5082	(49%)▲
DE	.3346	(51%)▲	.5965	(38%)▲	.3724	(64%)▲	.5163	(51%)▲
PSO	.3223	(46%)▲	.5965	(39%)▲	.3653	(61%)▲	.4936	(45%)▲
DE+LS	.3323	(51%)▲	.6017	(40%)▲	.3693	(63%)▲	.5009	(47%)▲
PSO+LS	.3143	(42%)▲	.5739	(33%)▲	.3604	(59%)▲	.5000	(47%)▲

Table 3: Test results with the 2007 dataset. Two retrieval performance measures are reported (Mean Average Precision, MAP, and Precision at ten, P@10). The values in each column are averaged over 5 runs. The symbols ▲ (▼) indicate a significant improvement (decrease) over both baselines ($BM25_{feed}$ and $\#PostsRanked_F$). Percentages of improvement were calculated w.r.t. $BM25_{feed}$, the best performing baseline.

query sets (max size: 5 queries). We would not be able to properly apply statistical test on the results. Furthermore, evaluating with at least 25 queries is a must in Information Retrieval. The complete set of training and test configurations can be found in Table 2. For each search algorithm, the training process (optimising MAP) was repeated 5 times and the scores of performance were averaged out.

In Table 3 and 4 we report the performance of the different optimisation algorithms. Two baseline methods ($BM25_{feed}$ and $\#PostsRanked_F$), which do not combine multiple sources of evidence, were also tested. $BM25_{feed}$ is a search algorithm that represents feeds as the concatenation of all their blog posts (title+body) and does retrieval using the effective BM25 matching function. By including this model into the evaluation we could quantify the improvement that can be obtained by using fusion methods instead of a basic full-text search on the feeds. $\#PostsRanked_F$ is a voting model with the post of the feed in the top 1000 of the retrieved post in a post-level index, every retrieved blog post from the feed is a vote for the relatedness between the query and the feed.

All fusion methods led to substantial improvements over the baselines (about 40%). The improvements reported correspond to the respective test folds (cross-

	2008a				2008b			
	MAP		P@10		MAP		P@10	
$BM25_{feed}$	.1567		.3560		.1469		.3160	
$\#PostsRanked_F$	.1293		.2959		.1611		.2840	
LS	.2332	(49%)▲	.3584	(1%)	.2413	(64%)▲	.4360	(38%)▲
DE	.2281	(45%)▲	.3632	(2%)	.2326	(58%)▲	.4240	(34%)▲
PSO	.2257	(44%)▲	.3544	(0%)	.2454	(67%)▲	.4096	(30%)▲
DE+LS	.2347	(50%)▲	.3608	(1%)	.2176	(48%)▲	.4160	(32%)▲
PSO+LS	.2223	(42%)▲	.3544	(0%)	.2422	(65%)▲	.4264	(35%)▲

Table 4: Test results with the 2008 dataset. Two retrieval performance measures are reported (Mean Average Precision, MAP, and Precision at ten, P@10). The values in each column are averaged over 5 runs. The symbols ▲ (▼) indicate a significant improvement (decrease) over both baselines ($BM25_{feed}$ and $\#PostsRanked_F$). Percentages of improvement were calculated w.r.t. $BM25_{feed}$, the best performing baseline.

validation). This means that the maximisation of the fitness function done with the training folds led to feature weights that worked very well for a separate test fold.

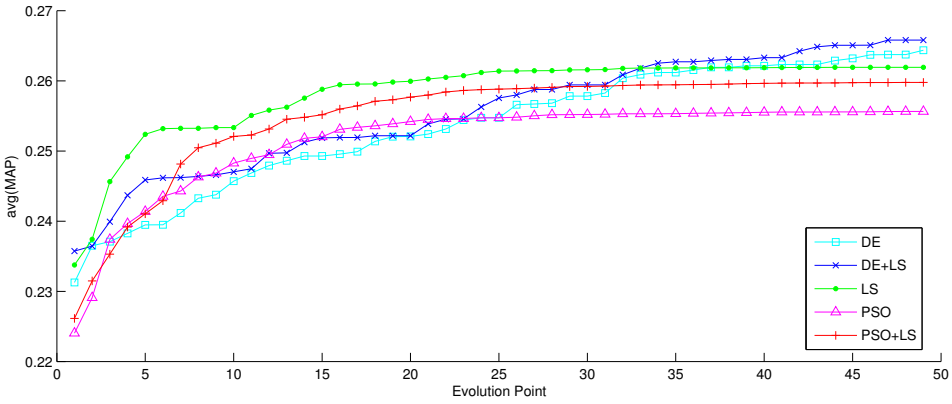
The $P@10$ improvements w.r.t. $BM25_{feed}$ obtained in the 2008*a* split (see Table 4) are rather low (and statistically insignificant). This might be due to the presence of some queries –e.g., *funny jokes*– that retrieve spam documents in the highest positions of the ranking[10]. Spam detection was out of the scope of this work and, therefore, we did not include any spam-based feature. From our analysis of the top 10 rankings of the 2008*a* split we hypothesise that spam filtering may lead to increased P@10. It can be observed how the voting model baseline performance is also very low in this case being affected by the same phenomenon. Spam detection for this task will be subject of further research.

The overall effectiveness of the fusion methods demonstrates that our features and optimisation algorithms are suitable in blog distillation. In terms of retrieval performance, we found no significant differences among the five fusion methods tested. There is no clear winner across all splits.

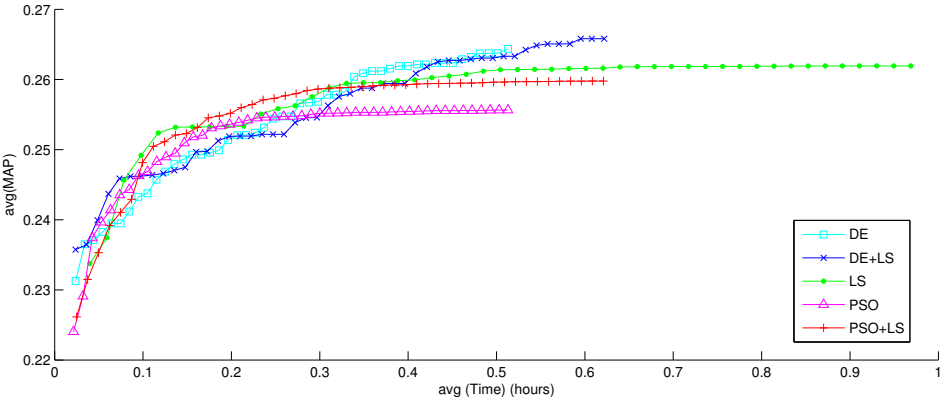
As argued above, efficiency is another important dimension in our study. Figure 2a plots the evolution of the optimisation in the training collection (average MAP) for the five algorithms. At the end of the execution, the best solution is given by the hybrid model DE+LS. DE yields the second highest MAP, LS goes in the third place, followed by PSO+LS and PSO. This is a fair comparison because all methods have the same number of computations of the fitness function –the most expensive operation– in every evolution point[11] (see Section 4.1).

[10]The TREC Blog Track contains spam documents (approx. 15%) [51].

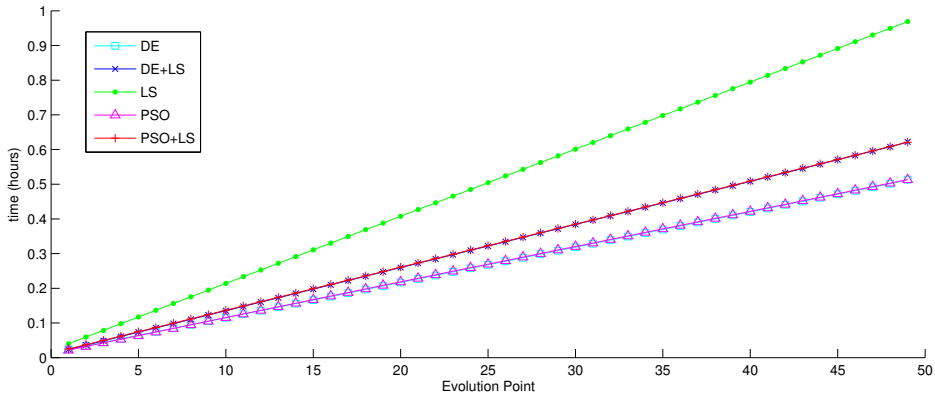
[11]Generation for the population-based methods or epoch in the case of LS.



(a)



(b)



(c)

Figure 2: Evolution of the different algorithms in training for the 2008b split (a) evolution point vs average MAP (b) average time in hours vs average MAP and (c) evolution point vs accumulated time in hours.

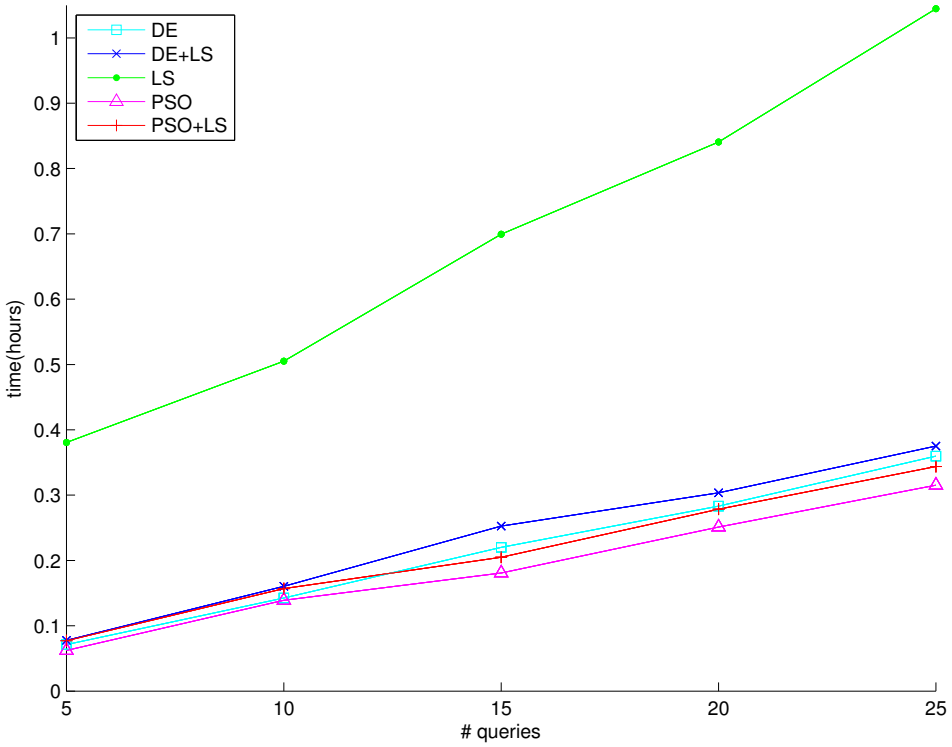


Figure 3: Evolution of the average execution time in hours for 50 generations when increasing the size of the problem when working with 5 to 25 queries in steps of 5.

At first sight, LS appears to be a good choice. It is the leading algorithm during a large part of the optimisation process. But LS does not explore appropriately the whole search landscape, as it strongly depends on the initial point. If the optimal solution is located at the limits of the search space then LS is much poorer than the global search methods. This has been empirically demonstrated by previous IR studies [29]. The second graph (2b) sheds light on a more important issue: time taken (instead of number of generations). When accounting for computational time, LS is not the dominant algorithm anymore. Observe (Figure 2b) that LS almost doubles the required time of execution of the evolutionary methods. The rest of the algorithms take 0.6 hours (or less) to finish the 50 generations, while LS takes almost 1 hour.

Although we implemented parallel code, LS has a synchronisation bottleneck (by design), which is the separation of Steps 1 and 2 (see Algorithm 1). This makes that the accumulated time per evolution point (bottom graph, Fig. 2c) grows much faster than the time for the evolutionary algorithms. Our hybrid proposals, which

	2007a		2007b	
	MAP	P@10	MAP	P@10
CMUfeedW	.3525	.5696	.3871	.5000
UMaTiPCSwGR	.2703	.5826	.2782	.4864
uogBDFeMNZP	.2577	.5565	.3284	.5045
kudsn	.2314	.5000	.2530	.4227
DUTDRun1	.2197	.4000	.2378	.3409
utblnrr	.2119	.4783	.2277	.4227
uams07bdtblm	.1271	.3130	.1954	.3000
TDWHU200	.0226	.0826	.0041	.0318

(a)

	2008a		2008b	
	MAP	P@10	MAP	P@10
KLEDistLMT	.3016	.4440	.3014	.4520
BM25LenNorm	.2632	.4040	.2499	.3880
uams08bl	.2627	.4120	.2649	.4280
cmuLDwikiSP	.2620	.4080	.3493	.4600
UMassBlog1	.2473	.3520	.2567	.4240
uogTrBDfeNWD	.2341	.3880	.2702	.4200
UBDist1	.2325	.3720	.2495	.3720
kudb	.2229	.3360	.2615	.3520
PermMeWhu	.1848	.3400	.1948	.2960
DUTIR08DRun1	.1579	.2960	.1621	.2240
FEEDKGP	.1528	.2680	.1550	.2680
feupbase	.1366	.2600	.1460	.2520

(b)

Table 5: Retrieval Performance Results (MAP and P@10) of TREC systems with the four query subsets.

apply LS to the best individual of each generation, are a good trade-off between effectiveness and computational effort.

Moreover, recent paradigms and technologies such as MapReduce, Hadoop or Apache Shark can be used in order to implement our solution when the vertical scalability limit is reached. For instance, in IR it is very common to use Hadoop in order to process the document collections and extract the features from the indices [55], allowing the indexing of big data scale collections. Furthermore, recent works [56, 57] tackled the derivation of the proposed methods into the MapReduce paradigm, facilitating the use of those distributed implementations, when required by the data size.

The presented fusion approaches are expected to be linear when increasing the size of the problem. For assessing this, we performed experiments varying the number of queries in the benchmark. The results are plotted in Figure 3. As expected, the evolution of the computational time for 50 generations reflects a linear behaviour.

4.4. Comparison with TREC systems

To put things in perspective, we compare now the retrieval performance of the fusion methods with the retrieval performance of the systems that participated in the TREC Blog Track competition. This comparison needs to be taken with care. On one hand, the 2007 participants did not have relevance judgements available

	2008a				2008b			
	MAP		P@10		MAP		P@10	
$BM25_{feed}$	.1567		.3560		.1469		.3160	
LS	.2688	(72%)▲	.3896	(9%)	.2550	(74%)▲	.4320	(37%)▲
DE	.2882	(84%)▲	.3952	(11%)	.2667	(82%)▲	.4352	(38%)▲
PSO	.2558	(63%)▲	.3608	(1%)	.2426	(65%)▲	.4128	(31%)▲
DE+LS	.2852	(82%)▲	.3856	(8%)	.2650	(80%)▲	.4320	(37%)▲
PSO+LS	.2433	(55%)▲	.3712	(4%)	.2643	(80%)▲	.4232	(34%)▲

Table 6: Test results with the 2008 dataset after incorporating the average retrieval performance of TREC systems. The symbols ▲ (▼) indicate a significant improvement (decrease) over the baseline ($BM25_{feed}$).

and, therefore, they could not apply any sort of train-test methodology[12]. On the other hand, many TREC systems incorporated evidence from a variety of sources (e.g., link analysis and spam filtering), and employed massive external datasets (e.g., Wikipedia). This luxury was not afforded to our fusion models because testing every single type of feature is out of the scope of this work.

In TREC 2007 (Table 3 vs Table 5a), the fusion models outperform all TREC systems (P@10). In terms of MAP, only the run from Carnegie Mellon (CMUfeedW) was better than the fusion models. This run made extensive use of external data (from the Wikipedia), which explains its increased recall of relevant feeds.

In TREC 2008 (Table 4 vs Table 5b), the fusion models rank in the middle of the pack. This might be due to the high variance in the degree of relevance of the feeds. The 2008 queries and relevance judgements were constructed in a way that blogs are assigned non-binary assessments. The human assessors were asked to label a feed as relevant when it contains enough on-topic posts. This loose notion of relevance potentially leads to relevant feeds that do not have a recurring principle interest in the topic area[13]. Some of our features directly estimate how recurrent the query topic is within the feed and, therefore, they might be unsuitable for low relevant feeds. This effect will be subject of further research.

Since our information fusion models can naturally encompass new types of evidence, it is possible to test what is the effect of including evidence from the TREC systems. As argued above, these systems incorporate signals that are complementary to ours. We made an additional experiment based on replacing the

[12]The 2008 participants did not have relevance judgements for the 2008 topics but they could use the 2007 topics for training.

[13]Instead, the 2007 assessors were explicitly told to make a judgement on whether the feed has a recurring principle interest in the topic area.

content-based features $RelBM25Feed\_Titles$, $RelBM25Feed\_Titles + Body$, and $RelBM25Feed\_Comments$ by a feature computed as the average score of relevance given to the feed by the TREC systems (after removing the best and the worst TREC systems). Table 6 shows the results of this experiment. Injecting evidence from the TREC systems makes that the information fusion methods are among the best performing systems. DE and DE+LS are the most dominant algorithms among our proposals. This demonstrates the good performance of DE when optimising real values [49].

5. Conclusions

In this paper we have approached Blog Distillation as an Information Fusion problem. The massive number of feeds in the blogosphere, the huge number of blog posts and comments, and the multiple signals available to estimate relevance turn this problem into a Big Data challenge. We have demonstrated that features of different nature can be combined in simple ways to estimate the relevance of a feed.

The high number of parameters and the size of the training sets demand solutions based on efficient optimisation. Therefore, we have compared the relative merits of different search algorithms, such as Line Search and population-based methods (Particle Swarm Optimisation and Differential Evolution). We have also designed novel hybrid methods that trade between computational effort and effectiveness. Not only we showed that the algorithms are efficient, but we also demonstrated that the resulting retrieval performance is competitive when compared to state-of-the-art methods. In summary, the main novelty of the paper is the use of population-based search algorithms and the defined hybrid versions with a local search method to optimise the parameter weights that are required for combining multiple types of evidence in blog distillation search. To the best of our knowledge, there is no previous work applying this methodology to this task, where combining effectively and efficiently different types of evidence (query-independent, query-dependent, blog-level, blog post-level, etc.) is critical.

In the near future, we plan to include additional optimisation algorithms and fusion models into our study. Another interesting line of future work consists of studying the interaction and dependencies among features. In this respect, we would like to incorporate a feature selection strategy to extract the most discriminative features and to skip weak features.

Acknowledgments

This work was supported by the *"Ministerio de Economía y Competitividad"* of the Goverment of Spain under the research projects TIN2012-33867 and TIN2013-40981-R.

References

[1] R. Santos, C. Macdonald, R. McCreadie, I. Ounis, I. Soboroff, Information Retrieval on the Blogosphere, Foundations and Trends in Information Retrieval 6 (2012) 1–125.

[2] K. Jost, M. J. Hipolit, Blog explosion, CQ Researcher 22 (2006) 505–528.

[3] C. Macdonald, I. Ounis, I. Soboroff, Overview of the TREC 2007 blog track, in: Proceedings of 16th Text Retrieval Conference (TREC 2007), National Institute of Standards and Technology, 2007.

[4] I. Ounis, C. Macdonald, I. Soboroff, Overview of the TREC 2008 blog track, in: Proceedings of 17th Text Retrieval Conference (TREC 2008), National Institute of Standards and Technology, 2008.

[5] J. L. Elsas, J. Arguello, J. Callan, J. G. Carbonell, Retrieval and feedback models for blog feed search, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, ACM, New York, NY, USA, 2008, pp. 347–354.

[6] Y. Lee, S. hoon Na, J. Kim, S. hyob Nam, H. young Jung, J. hyeok Lee, KLE at TREC 2008 blog track: Blog post and feed retrieval, in: Proceedings of 17th Text Retrieval Conference (TREC 2008), National Institute of Standards and Technology, 2008.

[7] J. Callan, Distributed information retrieval, in: W. Croft (Ed.), Advances in Information Retrieval, volume 7 of *The Information Retrieval Series*, Springer US, 2000, pp. 127–150.

[8] J. Seo, W. B. Croft, Blog site search using resource selection, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08, ACM, New York, NY, USA, 2008, pp. 1053–1062.

[9] P. Bailey, A. P. D. Vries, N. Craswell, I. Soboroff, Overview of the TREC 2007 enterprise track, in: Proceedings of 16th Text Retrieval Conference (TREC 2007), National Institute of Standards and Technology, 2007.

[10] C. Macdonald, I. Ounis, Key blog distillation: Ranking aggregates, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08, ACM, New York, NY, USA, 2008, pp. 1043–1052.

[11] C. Macdonald, I. Ounis, Voting for candidates: Adapting data fusion techniques for an expert search task, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06, ACM, New York, NY, USA, 2006, pp. 387–396.

[12] R. Dueñas Fernández, J. D. Velásquez, G. LHuillier, Detecting trends on the Web: A multidisciplinary approach, Information Fusion 20 (2014) 129–135.

[13] B. Croft, Combining approaches to information retrieval, in: B. Croft (Ed.), Advances in Information Retrieval, volume 7 of *The Information Retrieval Series*, Springer US, 2000, pp. 1–36.

[14] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, C. Burges, Optimisation methods for ranking functions with multiple parameters, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06, ACM, New York, NY, USA, 2006, pp. 585–593.

[15] T.-Y. Liu, Learning to rank for information retrieval, Found. Trends Inf. Retr. 3 (2009) 225–331.

[16] R. Herbrich, T. Graepel, K. Obermayer, Large margin rank boundaries for ordinal regression, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000, pp. 115–132.

[17] K. Crammer, Y. Singer, Pranking with ranking, in: Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 641–647.

[18] R. Herbrich, T. Graepel, K. Obermayer, Support vector learning for ordinal regression, in: Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470), volume 1, 1999, pp. 97–102.

[19] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, J. Mach. Learn. Res. 4 (2003) 933–969.

[20] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 89–96.

[21] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, W.-Y. Ma, Frank: A ranking method with fidelity loss, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 383–390.

[22] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, H. Li, Query-level loss functions for information retrieval, Inf. Process. Manage. 44 (2008) 838–855.

[23] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to rank: From pairwise approach to listwise approach, in: Proceedings of the 24th International Conference on Machine Learning, ICML '07, ACM, New York, NY, USA, 2007, pp. 129–136.

[24] Y. Lan, T.-Y. Liu, Z. Ma, H. Li, Generalization analysis of listwise learning-to-rank algorithms, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 577–584.

[25] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, H. Li, Listwise approach to learning to rank: Theory and algorithm, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 1192–1199.

[26] D. Bollegala, N. Noman, H. Iba, Rankde: Learning a ranking function for information retrieval using differential evolution, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 1771–1778.

[27] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.

[28] J. M. Chenlo, A. Hogenboom, D. E. Losada, Sentiment-based ranking of blog posts using rhetorical structure theory, in: NLDB, 2013, pp. 13–24.

[29] J. Parapar, M. Vidal, J. Santos, Finding the Best Parameter Setting: Particle Swarm Optimisation, in: Proceedings of the 2nd Spanish Conference on Information Retrieval (CERI 2012), Valencia (Spain), 2012, pp. 49–60.

[30] G. Zeng, Y. Jiang, A modified PSO algorithm with line search, in: Proceedings of 2010 International Conference on Computational Intelligence and Software Engineering (CiSE), 2010, pp. 1–4.

[31] M. Melanie, An Introduction to Genetic Algorithms, Computers Mathematics with Applications 32 (1996) 133.

[32] F. Herrera, M. Lozano, Fuzzy Genetic Algorithms: Issues and Models, Technical Report, University of Granada, 1999.

[33] G. Salton, The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[34] L. Polanyi, A. Zaenen, How Okapi came to TREC, in: TREC: Experiments and Evaluation in Information Retrieval, The MIT Press, 2005, pp. 287–299.

[35] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to information retrieval, ACM Trans. Inf. Syst. 22 (2004) 179–214.

[36] J. Parapar, J. López-castro, A. Barreiro, Blog posts and comments extraction and impact on retrieval effectiveness, in: Proceedings of the 1st Spanish Conference on Information Retrieval , (CERI 2010), Madrid (Spain), 2010, pp. 5–16.

[37] J. M. Chenlo, J. Parapar, D. E. Losada, Comments-oriented query expansion for opinion retrieval in blogs, in: C. Bielza, A. Salmern, A. Alonso-Betanzos, J. Hidalgo, L. Martnez, A. Troncoso, E. Corchado, J. Corchado (Eds.), Advances in Artificial Intelligence, volume 8109 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, pp. 32–41.

[38] I. Ruthven, M. Lalmas, A survey on the use of relevance feedback for information access systems, Knowl. Eng. Rev. 18 (2003) 95–145.

[39] V. Lavrenko, W. B. Croft, Relevance based language models, in: Proc. of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR'01, ACM, New York, NY, USA, 2001, pp. 120–127.

[40] J. Seo, J. Jeon, High precision retrieval using relevance-flow graph, in: Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, ACM, New York, NY, USA, 2009, pp. 694–695.

[41] J. M. Chenlo, D. E. Losada, Combining Document and Sentence Scores for Blog Topic Retrieval, in: Proceedings of the 1st Spanish Conference on Information Retrieval , (CERI 2010), Madrid (Spain), 2010.

[42] J. Parapar, J. López-Castro, A. Barreiro, Blog snippets: A comments-biased approach, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, ACM, New York, NY, USA, 2010, pp. 711–712.

[43] D. G. Luengerber, Linear and nonlinear programming, Addison Wesley, 1984.

[44] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948.

[45] J. Kennedy, R. Eberhart, Swarm intelligence, Morgan Kaufmann Publishers Inc., CA, USA, 2001.

[46] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[47] K. Price, R. Storn, J. Lampinen, Differential Evolution. A Practical Approach to Global Optimization, Springer - Natural Computing Series, 2005.

[48] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.

[49] V. Feoktistov, Differential Evolution: In Search of Solutions, Springer, NY, 2006.

[50] D. Whitley, V. Gordon, K. Mathias, Lamarckian evolution, the Baldwin Effect and function optimization, Lecture Notes in Computer Science 866 (1994) 6–15.

[51] C. Macdonald, I. Ounis, The TREC Blogs06 Collection: Creating and Analysing a Blog Test Collection, Technical Report, 2006. Technical Report No.: TR-2006-224.

[52] C. Fox, A stop list for general text, SIGIR Forum 24 (1989) 19–21.

[53] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computation, volume 1, 2000, pp. 84–88 vol.1.

[54] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

[55] R. Mccreadie, C. Macdonald, I. Ounis, Mapreduce indexing strategies: Studying scalability and efficiency, Inf. Process. Manage. 48 (2012) 873–888.

[56] C. Zhou, Fast parallelization of differential evolution algorithm using MapReduce, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, ACM, New York, NY, USA, 2010, pp. 1113–1114.

[57] A. McNabb, C. Monson, K. Seppi, Parallel PSO using MapReduce, in: IEEE Congress on Evolutionary Computation, 2007. CEC 2007, 2007, pp. 7–14.