# Pragmatic Ontology Evolution: Reconciling User Requirements and Application Performance

Francesco Osborne(✉) and Enrico Motta

Knowledge Media Institute, The Open University,
Milton Keynes MK7 6AA, UK
{francesco.osborne,enrico.motta}@open.ac.uk

**Abstract.** Increasingly, organizations are adopting ontologies to describe their large catalogues of items. These ontologies need to evolve regularly in response to changes in the domain and the emergence of new requirements. An important step of this process is the selection of candidate concepts to include in the new version of the ontology. This operation needs to take into account a variety of factors and in particular reconcile user requirements and application performance. Current ontology evolution methods focus either on ranking concepts according to their relevance or on preserving compatibility with existing applications. However, they do not take in consideration the impact of the ontology evolution process on the performance of computational tasks – e.g., in this work we focus on instance tagging, similarity computation, generation of recommendations, and data clustering. In this paper, we propose the *Pragmatic Ontology Evolution (POE)* framework, a novel approach for selecting from a group of candidates a set of concepts able to produce a new version of a given ontology that (i) is consistent with the a set of user requirements (e.g., max number of concepts in the ontology), (ii) is parametrised with respect to a number of dimensions (e.g., topological considerations), and (iii) effectively supports relevant computational tasks. Our approach also supports users in navigating the space of possible solutions by showing how certain choices, such as limiting the number of concepts or privileging trendy concepts rather than historical ones, would reflect on the application performance. An evaluation of POE on the real-world scenario of the evolving Springer Nature taxonomy for editorial classification yielded excellent results, demonstrating a significant improvement over alternative approaches.

**Keywords:** Ontology evolution · Domain ontologies · Bibliographic data Scholarly data · Scholarly ontologies

## 1 Introduction

Increasingly, organizations are adopting ontologies to describe their large catalogues of items. Indeed, ontologies have proved to be very useful in the context of a variety of tasks [1], including the integration of data from different sources, domain reasoning, classification [2], generation of recommendations [3], cluster analysis [4], community detection [5], sentiment analysis, forecasting [6], and others. Naturally, ontologies need to be regularly maintained and need to evolve according to changes in the domain or new requirements from users or applications [7]. This process is called *ontology evolution* and it is a critical part of the ontology lifecycle. While the literature proposes a variety of frameworks for ontology evolution [8–11], essentially most agree on three fundamental steps in the process: (i) detection of the need for the evolution, (ii) identification of candidate changes, and (iii) validation and assessment of these changes, to ensure that the resulting ontology satisfies the given needs.

Hence, in the first instance, the evolved ontology normally has to comply with a set of requirements, defined to ensure that the ontology remains compatible with the current workflow and usable by the relevant stakeholders.

In the second instance, it is crucial to take into account the impact of the ontology evolution process on relevant applications. Ontologies are often used to enable semantic approaches to data mining, information filtering, trend detection, and other tasks [12], whose performance needs to be taken in consideration when creating a new version of the ontology. Crucially, user needs and applications performance are sometimes in opposition. For example, a very comprehensive representation of items and their features would generally improve the performance of a recommender system, but users may prefer a less complex representation that it is easier to browse, memorize, maintain, and incorporate in their workflow.

In the third instance, domain experts may have preferences about which concepts to privilege that should be considered in the process. For example, they may decide to privilege concepts which are currently trendier rather than historical ones, or those that are more represented in their internal catalogue, rather than considering the full domain.

Finally, users need to be able to understand why a certain concept was selected or discarded and how this relates to the requirements, the user preferences, and the ontology support for some computational tasks.

The motivating scenario for this work concerns the evolution of the internal taxonomy at Springer Nature, which is used for classifying books, journals, and other editorial products. Since this taxonomy is used by a lot of different users and software systems, the evolution process needs to take in consideration both user needs and the impact on applications. For instance, a recommender system for suggesting editorial products described by an ontology [13] would perform differently according to the ontology that it is using. In addition, the process need to be transparent, so that every change can be justified in light of these factors.

Current solutions are not easily applicable to this problem. Most of the methods for selecting the concepts to be included in an evolving ontology address this task by ranking concepts according to a weight derived from information retrieval metrics [14, 15], list of words [16], or online ontologies [11]. These solutions have the advantage of being generic, but present two significant limitations: (i) they do not assess the impact of the new version of the ontology on the performance of the relevant applications, and (ii) they ignore concept synergy, by weighting the relevance of single concepts rather than the overall impact of a combination of concepts. Some approaches do focus on preserving consistency between the ontology and the dependent applications [17–21], however they do not consider the effect of the changes on the performance of computational tasks.

In this paper, we propose the *Pragmatic Ontology Evolution* (POE) framework, a novel approach for selecting from a group of candidates a set of concepts able to produce an ontology that (i) is consistent with the given requirements, (ii) is parametrised with respect to a number of dimensions (e.g., topological considerations), and (iii) supports effectively relevant computational tasks, such as instance tagging, similarity computation, generation of recommendations, and data clustering. POE supports users in navigating the space of possible solutions by showing how certain choices, such as limiting the number of concepts or privileging trendy concepts rather than historical ones, would reflect on the application performance. It also makes it easy to explain why a certain concept was included in the ontology on the basis of its contribution to the performance of a specific task. Finally, it selects the new concepts not only according to individual weights, but also considering their synergy with other concepts.

The rest of the paper is organized as follows. In Sect. 2, we will present a motivating scenario involving the evolution of an editorial taxonomy at Springer Nature. In Sect. 3, we will review the literature regarding ontology evolution and, in particular, the selection of candidate concepts. In Sect. 4, we will discuss POE in details and in Sect. 5, we will evaluate it on a dataset of 1,218 Springer Nature books. Finally, in Sect. 6, we summarize the main conclusions and outline future directions of research.

## 2 Motivating Scenario: Evolving Springer Nature Market Codes

Springer Nature (SN) is one of the major academic publishing companies and has a vast catalogue of books, journals, and conference proceedings. Like other companies in this space, it has its own editorial classification system, called *Product Market Codes* (PMC). PMC is a taxonomy of research fields that is used to tag editorial items with relevant topics, e.g., "Artificial Intelligence" or "Software Engineering". The resulting metadata are then used for a variety of tasks, such as improving the discoverability of products in digital and physical libraries, supporting marketing decision, and detecting research trends.

It is crucial to keep PMC up to date with the evolution of the research landscape at the right level of granularity. This is particularly challenging in the field of Computer Science, where new areas evolve constantly and taxonomies tend to become obsolete very quickly [22]. In the context of the collaboration between The Open University and Springer Nature [2, 13], we focused on the issue of supporting the evolution of the Computer Science portion of PMC, concentrating in particular on some branches that had become obsolete.

This work builds on our earlier research, which has produced new methods able to generate automatically taxonomies of research areas through large scale-mining of scholarly data. In particular, by applying the Klink-2 algorithm [22] on the Rexplore dataset [23], we generated the Computer Science Ontology (CSO) [24], a large-scale ontology of research topics in Computer Science, which includes about 26K topics linked by about 226K semantic relationships. CSO powers two tools used by SN for tagging and recommending books: Smart Topic Miner [2] and Smart Book Recommender [13].

In accordance with the requirements provided by SN publishing editors, we focused on the evolution of the branches under five concepts of the original PMC taxonomy ("I21017-Artificial Intelligence (incl. Robotics)", "I14029-Software Engineering", and three others – see details in Sect. 5) that we mapped to nine CSO concepts (in the given example "Artificial Intelligence", "Robotics", "Software Engineering", and "Software Design"). We then extracted all their sub-concepts producing 2,451 candidate concepts. However, producing a new version of PMC with all of them, would cause the Computer Science portion of PMC to grow from 89 to 2,540 concepts. This is unfeasible for a variety of pragmatic reasons, including the fact that many books are still manually tagged and curated by editors. We thus needed to find a solution to the evolution of PMC, which ensured that it remained under a certain size. It was also crucial that the new version of the ontology would support effectively tasks such as generation of recommendations, data clustering, and so on. Finally, we would need to be able to produce a justification for the inclusion or the exclusion of a research topic.

This is a typical real-world case in which the first two steps of the ontology evolution process, identifying the need for changes and producing candidate concepts, are relatively easy, since it exists a clear need (new fields in Computer Science are missing) and we already have a good selection of candidate concepts in CSO. On the contrary, there was no clear solution for selecting a set of concepts that would comply with the requirements and support relevant applications.

## 3 Related Work

Most of the ontology evolution frameworks [7–10] include a phase that regards the verification and selection of candidate changes or concepts to be included in the new version of the ontology. This step is labelled "change validation phase" in the framework of Stojanovic [8], "verification and approval" in Klein and Noy [9], "accepting and rejecting changes" in Noy [10], and it is split in two different phases labelled "validating changes" and "assessing the evolution impact" in the ontology evolution cycle proposed by Zablith [7].

Traditionally, the candidate changes are validated at three different levels [7]: (i) *formal properties-based validation*, which uses formal techniques to preserve the consistency and coherence of the ontology, (ii) *domain base validation*, which exploits domain information to assess the relevance of the candidate changes, and (iii) *application and usage impact*, which measures the effects of the changes on data instances, dependent ontologies, and relevant applications [25]. POE works at the second and third levels, since it assesses the importance of concepts within a certain domain and it evaluates the effect of alternative ontologies on computational tasks.

Approaches to *domain base validation* can be classified according to their focus, which can be either on domain relevance [14–17, 26] or correctness [27]. Text2Onto [14], a well-known system for ontology learning, falls in the first category, since it weights the relevance of the candidate concepts by mean of information retrieval measures, such as Relative Term Frequency (RTF), TF-IDF, and the C-value/NC-value method. SPRAT [15], a tool for automatic pattern-based ontology population, also uses TF-IDF to select the relevant terms that should be included in the ontology. Similarly to POE, they both focus on the inclusion of concepts or terms rather than entire statements. The DINO Framework [16], assesses the relevance of a set of candidate triples according to their Levenshtein distance from a set of wanted or unwanted words, specified by domain experts. The Evolva framework [11] measures the relevance of a statement by generating its ontological context from a set of online ontologies and comparing it to the evolving ontology. The DINAMO-MAS system [26] assesses relationships between terms by means of a confidence score that takes in consideration their lexico-syntactic patterns. Some other systems focus on assessing the correctness of statements. For instance, Sabou et al. [27] verify the correctness of the link between two concepts by exploiting the path connecting the concepts in online ontologies. Similarly to these solutions, POE aims to find the best set of concepts to be included in an evolved ontology. However, it also consider application performance and concept synergy.

Some other approaches focus on assessing the impact of evolution on data instances [28, 29], applications [17–21, 30], and dependent ontologies [25]. Because of lack of space, we will focus our review on the first two categories.

Qin and Atluri [28] propose a method to define and preserve the structural and semantic validity of data instances that are described by an evolving ontology. Similarly, Hartung et al. [29] introduce a generic framework for the study of the evolution of ontologies and ontology-related mappings. We also take into consideration instances and their mapping, but rather than checking their validity, we focus on the impact of their representations on the relevant tasks.

Several approaches address the impact of the resulting ontology on dependent applications, however they focus mainly on preserving consistency and compatibility. For instance, Huang and Stuckenschmidt [17] present MORE, a system that uses temporal logic to detect the consequences of changes. Xuan et al. [18] introduce the floating version model, which preserves compatibility by not allowing a new version of the ontology to falsify axioms that were previously true. Wang et al. [19] propose another technique to maintain the consistency of dependent applications and suggest resolution strategies. Liang et al. [20] present a system that analyses the queries submitted by dependent applications, detects if the relevant entities where changed during the evolution process, and repairs broken queries. Similarly, Kondylakis and Plexousakis [21] propose a formal approach for identifying the impact of ontology evolution on queries and easing query migration. Finally, Groß et al. [30] introduce an approach for measuring the stability of a ontology and show how ontology evolution affected the level of significance of functional enrichment analyses in Biology. Differently from all these systems, POE focuses on the performance of dependent computational tasks rather than on consistency and compatibility, and aims to generate an ontology that can effectively support these tasks.

## 4 The POE Framework

### 4.1 Overview of POE

The Pragmatic Ontology Evolution (POE) framework was designed to produce an ontology that complies with the given requirements and performs well on some input tasks, as well as supporting users in exploring the space of solutions. POE takes as input (i) an ontology, (ii) a collection of instances that could be described by the concepts in the ontology, (iii) a set of additional candidate concepts (and their relationships with existing concepts), (iv) a set of requirements, (v) one or more tasks, and, optionally, (vi) four additional parameters defining user preferences. It then finds the combination of candidate concepts that generates the representation of the instances which performs best on the given tasks by first searching in the space of four parameters and then applying a variation of Recursive Feature Elimination [31]. Finally, it returns: (i) a new version of the ontology that complies with the input requirements and effectively supports the relevant tasks, and (ii) a number of statistics that allow users to assess the effect of their preferences (e.g., privileging conservative or novel concepts) on the tasks.

In the PMC scenario, the input ontology is the portion of PMC covering the field of Computer Science, while the instances are the metadata of books published by SN in recent years and tagged with PMC concepts. The set of candidate concepts was built by mapping the PMC concepts that needed to be enriched to relevant concepts in CSO and then selecting all their sub-concepts, as discussed in Sect. 2. The mapping was done semi-automatically by generating candidate mapping with statistical heuristics from Klink-2 [22] and then revising them with the help of SN editors, as described in [2]. This operation yielded 2,451 candidate concepts.

The POE framework is structured in two main steps:

**Parameter Optimization.** It tests different combinations of four parameters (using grid search) to weigh the candidate concepts. For each combination, it produces an ontology that complies with the requirements, it annotates the instances with it, and it measures the performance of this representation on the tasks. Finally, it returns a ranked list of parameter combinations.

**Recursive Concept Elimination.** It uses the best parameter combination from previous steps to generate an ontology and applies on it a variation of the Recursive Feature Elimination to iteratively eliminate the least important concepts, until the desired number of concepts is reached.

POE allows users to set *three kind of requirements*: (1) the maximum number of concepts in the ontology, (2) the minimum number of concepts in a branch, and (3) the maximum number of concepts in a branch. Being able to control the dimension of branches is important to produce structurally balanced ontologies. POE also allows the users to define or restrict (within a range) *four parameters* that control the ranking of the candidate concepts.

POE can be used with any task that uses an ontology-derived representation of the instances and whose performance can be evaluated according to an objective metric. In particular, in the current prototype we support four tasks: *instance tagging*, *similarity computation*, *generation of recommendations*, and *data clustering*.

In what follows, we will first discuss the basic functions of POE, i.e., the generation of an ontology from a set of parameters (Sect. 4.2), and the evaluation of a ontology on a task (Sect. 4.3). We will then address the two main steps of the POE framework that employ these functionalities: parameter optimization in Sect. 4.4, and Recursive Concept Elimination in Sect. 4.5.

### 4.2 Topic Ranking

In this phase, we consider the task of selecting a number of concepts to update an ontology as a ranking problem, coherently with the state of the art (e.g., [11, 14–16]). We thus want to assign a weight to every concept and then update the input ontology with the first *n* concepts that comply with the requirements.

A typical way to do so is assessing a concept importance according to how frequently it is represented in the instances. Intuitively, a concept that is often needed to describe the instances should receive a higher weight than a rarer one. Indeed, previous literature showed that term frequency and TF-IDF perform quite well on this task [14, 15]. We believe however that is possible to have a more comprehensive treatment of this challenge by taking in consideration a number of additional factors. In particular, here we consider four dimensions that can influence the value of a concept in the new ontology and the strategy for mapping it to the instances.

**Semantics.** As already mentioned, a purely syntactical solution to weigh concepts is to use the frequency of their label in the instances. For example, given the concept *temporal logic*, we could weigh it according to the number of books that contains in the title, abstract, or keyword field the string "temporal logic". Alternatively, we could take a more semantic approach and associate to a concept each instance that contains the label of the concept or of any of its sub-concepts. For example, we could map the concept *temporal logic* to each book that contains one of the alternative labels (e.g., "temporal logics") or sub-concepts (e.g., "temporal operators") in the CSO ontology. This technique has been applied with good results in a variety of fields, such as automatic classification of proceedings [2], technology forecasting [6], recommender systems [3, 13], community detection [5], and others.

**Temporal Dimension.** It is also useful to consider when the instances were produced. In the scenario of academic publishing, considering recent instances would prioritise the trendiest research topics, which may keep growing and become more popular in the future. However, focusing too much on recent instances, may exclude some significant historical concepts that are still important and may risk prioritising concepts that are experiencing only a transient burst of popularity.

**Internal Versus External Instances.** The instances can either derive from the catalogue of the organization that has adopted the ontology (e.g., SN books in Computer Science) or they could be generic ones (e.g., all available books in Computer Science). In the first case, the selected concepts will acquire the same biases of the internal dataset. The resulting ontology will be tailored to those specific instances, but may exclude significant concepts that are currently under-represented in the catalogue. Therefore, a company that wants to expand its catalogue and cover new fields may prefer to consider all available instances, while one that is not interested in doing so, may decide to produce a more internally-tailored ontology.

**Structural Considerations.** Considering only the weight of single concepts may exclude some concepts that are less represented in the instances, but act as good branching point in the ontology and keep the structure easy to browse and explore. Therefore, in some cases it may be advisable to include concepts that are useful from a structural standpoint, even if they appear less frequently in the instances.

We believe that it is useful to take in consideration each of these dimensions when ranking concepts. Therefore, POE takes as input four parameters that can be tuned by the user or optimized on a certain task:

- $\alpha$ (0−1). It controls whether POE uses the syntactic method, the semantic method, or a combination of the two for mapping concepts to instances. If $\alpha = 0$, it will use only the label of a concept, with $\alpha = 1$ it will consider all the sub-topics, otherwise it will use a weighted average.
- $\beta(0-1)$. It controls whether the weight will be computed only on instances from an internal dataset or if it will consider also external entities. If $\beta = 0$, POE will use only the internal instances, with $\beta = 1$ only external ones, otherwise it will use a weighted average.
- $\gamma\,(0-1)$. It modulates the importance of the most recently created instances on the weight. If $\gamma = 0$, POE will weight more recent instances, with $\gamma = 1$ the time dimension will not matter, otherwise it will use a weighted average.
- $\delta$ (True, False). It controls whether POE will try to recover structurally important concepts. In the current implementation, a concept is considered structurally important if it has at least three sub-concepts that were selected.

The weight of each concept is computed with the following formula.

$$\left(\left(\log\sum_{y=f}^{l} si_y^{w_y}\right)\beta+\left(\log\sum_{y=f}^{l} se_y^{w_y}\right)(1-\beta)\right)\alpha+\left(\left(\log\sum_{y=f}^{l} fi_y^{w_y}\right)\beta+\left(\log\sum_{y=f}^{l} fe_y^{w_y}\right)(1-\beta)\right)(1-\alpha)$$

Where $si_y$, $se_y, fi_y, fe_y$ are respectively, for a given year $y$, the semantic frequency in the internal dataset, the semantic frequency in the external dataset, the syntactic frequency in the internal dataset, and the syntactic frequency in the external dataset; $f$ and $l$ are the first and last year of the analysed period; and $w_y = \frac{1}{(l-y+1)^{2\gamma}}$.

After ranking the concepts, POE selects the first $n$ concepts that comply with the input requirements. The POE framework can adopt any kind of requirements that can be automatically verified by analysing the set of candidate concepts. In the current prototype we take in consideration the minimum and maximum number of concepts for each branch. POE enforces this requirements by first populating each branch with the minimum number of concepts and then inserting the remaining concepts in the branch that are still available until the maximum number of concepts is reached. If $\delta$ is true, POE also checks for structurally important concepts and inserts them in place of the ones with lowest weights. Finally, it creates a new version of the ontology which incorporates the selected concepts.

It is also possible to define a list of invalid topics that will not be considered during the selection phase. This option will be used during the Recursive Concept Elimination (Sect. 4.5) to exclude topics that do not perform well on the tasks.

The approach described in this section can be used on its own in alternative to generic methods [14, 15]. The main advantage is that it allows users to explore the space of solutions, possibly with the support of domain experts, and understand how different combination of parameters impact on the resulting ontology. However, it is difficult even for human experts to assess how a new ontology will affect applications. For this reason, we want to take a further step: evaluate the alternative ontologies on the input tasks and suggest the one that yields the best performance.

## 4.3 Evaluating a Candidate Ontology on a Task

POE evaluates an ontology on some computational tasks by (i) using the ontology for generating a representation of the instances, (ii) running the input tasks on this representation, and (iii) evaluating the performance with the relevant metrics. The instances are represented as a vector in which the elements correspond to the concepts in the ontology and the values weigh the importance of a concept. In the case of PMC, we used the Smart Topic API [13] for representing books as a vectors of research topics in which each topic is assigned a value equal to the number of chapters in which it appears. This is a convenient representation that can support several tasks. The Smart Topic API is a service developed in collaboration with Springer Nature for tagging publications with ontology concepts. It is described in details in [2, 13].

While some tasks (e.g., instance tagging) can be evaluated using simple metrics (e.g., percentage of instances covered), others require a ground truth. For instance, evaluating the performance of a clustering algorithm would usually require a correct set of clusters to compare against. In some cases, such as in the PMC scenario, it is quite expensive to produce a specific gold standard for each task. Therefore, we address this issue by adopting a ground truth ontology that includes all candidate concepts and can be used with every task. The intuition is that we want to select a candidate ontology including no more than *n* concepts that would perform as well as possible as the full ontology. In the case of PMC, we want to produce an ontology of about 120–200 concepts that can perform as closely as possible to the version which includes all 2,451 candidate concepts from CSO. In the following, we will refer to the candidate ontology as $O_c$ and to the full ontology, which serves as ground truth, as $O_f$.

It is important to note that if the task in consideration is sensitive to irrelevant or redundant features, the ground truth ontology needs to contain valid concepts and to have been previously evaluated. This is indeed the case with CSO, which was previously tested on several tasks [24], including automatic tagging of scientific publications [1], recommendation generation [2], clustering [5], and technology forecasting [6]. Alternatively, we suggest to pre-filter the candidate concepts [16] or to generate a task-specific gold standard.

The current POE prototype implements four tasks that were developed for the PMC scenario. The implementation of a new task is straightforward since it simple requires to define a representation of the instances, run the task on them, and evaluate the results with a relevant metric. If the input includes several tasks, their overall performance is computed as the average of the resulting metrics.

We will now discuss these tasks and their evaluation.

### 4.3.1 Instance Tagging

As first task, we consider the automatic tagging that associates each instance to a vector of concepts (via the Smart Topic API [13]). The candidate ontology should enable to generate a relatively granular representation of all the instances. Therefore, we evaluate this task by computing the percentage of instances that are covered by the ontology. Naturally, the definition and quality of the coverage varies according to the scenario and the domain. In the case of PMC, it is important to associate each book to a minimum number of topics, so that they can be browsed and searched with a good granularity. Furthermore, the main topics have to be fairly representative and not appear only in few chapters. We thus consider covered a publication that is associated with at least three concepts that are present in at least three chapters.

### 4.3.2 Similarity Computation

Computing the similarity of a set of items is a common task that supports more complex tasks such as record linkage, clustering, and so on. We evaluate this task by computing the cosine similarity of each couple of instances according to both $O_c$ and $O_f$, and then calculating their mean root-mean-squared error.

$$similarity\_performance = 1 - \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\cos(\widehat{c_\imath},\widehat{c_J}) - \cos\left(\widehat{f_\imath},\widehat{f_J}\right)\right)^2}$$

Where $\cos(\widehat{v_1},\widehat{v_2})$ is the cosine similarity between vectors $\widehat{v_1}$ and $\widehat{v_2}$, $\widehat{c_\imath}$ is the vector of instance $i$ produced with the candidate ontology, $\widehat{f_\imath}$ is the vector of instance $i$ produced with the full ontology, and $n$ is the total number of instances. When the result is near 1 the two ontologies are yielding similar results and thus the candidate ontology is performing well.

### 4.3.3 Generation of Recommendations

Today several recommender systems use ontologies for enhancing semantically the representation of items or users [3]. In particular, content-based recommenders use feature representations of items to suggest other items that possess similar characteristics. This is the case of *Smart Book Recommender* [13] which suggest SN books relevant to a certain conference.

We generate for each instance, say *I*, a ranked list of recommendations composed by the 100 instances most similar to *I*, according to both $O_c$ and $O_f$. This is realized by computing the cosine similarity of the vector representations derived from the two ontologies. We then assess the agreement of the lists produced by the two ontologies using the Spearman's rank correlation coefficient, a standard metric for evaluating recommender systems. The Spearman's coefficient between two variables equals to the

Pearson correlation between the rank values of those two variables, and it is used when it is important to compare the order of items in a list. It varies between −1 and 1, with 1 (or −1) indicating that the two list exhibit a perfect correlation and 0 indicating that the order of two list is not correlated at all. The performance of $O_c$ on this task is measured according to the following formula:

$$recommender\_performance = \frac{1}{n}\sum_{i=1}^{n}\frac{cov(rc_i, rf_i)}{\sigma_{rc_i}\sigma_{rf_i}}$$

Where $\sigma_{rc_i}$ and $\sigma_{rf_i}$ are the standard deviations of the ranked list of items according to $O_c$ and $O_f$, and $cov(rc_i, rf_i)$ is the covariance of the ranked lists.

### 4.3.4 Clustering

Cluster analysis is a powerful tool for exploring trends, generating analytics, and informing marketing and political decisions. We first cluster the instances according to both ontologies by using the K-Means++ algorithm and then compare the results with the Rand index, which is a measure of the similarity between two sets of clusters. The Rand index varies between 0 and 1, with 1 indicating that the data are clustered in the same way and 0 indicating that the cluster sets are completely dissimilar.

$$clustering\_performance = \frac{a_i + b_i}{\binom{n}{2}}$$

Where $a_i$ is the number of pairs of instances that are in the same cluster both in the cluster set of $O_c$ and in the cluster set of $O_f$, and $b_i$ is the number of pairs that are in different clusters.

## 4.4 Parameter Optimization

Parameter optimization is the first step of the POE approach. In this phase, POE executes a grid search on the space of the four parameters described in Sect. 4.2, produces a candidate ontology for every combination of parameters, and ranks them according to their performance on the tasks, as illustrated in Sect. 4.3. The ontology that performs best is the advisable solution in the space of parameters.

The result of this phase can be used for exploring the space of solutions and assessing the effect of the parameters on the ability of an ontology to perform certain tasks. A simple way to do so is testing if there is any correlation between a parameter and the performance. For instance, Fig. 1 shows the relation between two parameters and the performance obtained on the generation of recommendations task (Sect. 4.3.3) when representing 718 SN books in the 2012–2014 period with the ontology produced by including 40 additional topics to PMC. $\alpha$ is directly correlated with the recommender performance, yielding a Pearson correlation coefficient of 0.69 ($p < 0.0001$). It thus seems that mapping instances with the semantic approach works better when optimizing the ontology for this task. Although, it is interesting to notice that the best results are obtained when $0.5 \leq \alpha \leq 0.75$, therefore a purely semantic approach may be counterproductive. Conversely $\beta$ exhibits a mild inverse correlation with the performance, yielding a Pearson correlation coefficient of −0.36 ($p < 0.0001$). This indicates that preferring the instances from the internal dataset tends to produce a superior result on this task.

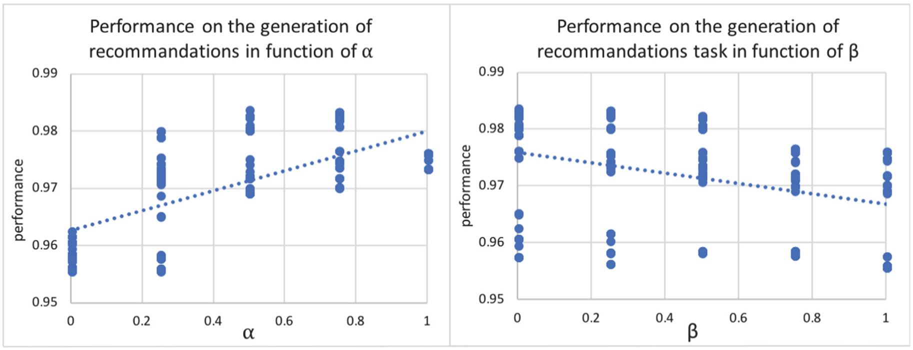

**Fig. 1.** Performance on the generation of recommendations task in function of α and $\beta$.

## 4.5 Recursive Concept Elimination

The previous step can outperform some more basic methods (see Sect. 5), but still suffers from two main limitations. First, the optimization was limited to the space of parameters, therefore a better solution may exist outside this space. Secondly, the typical strategy of assigning weights to single concepts does not take into consideration concept synergy. Conversely, it is possible that even if concept $C_1$ has lower weight than $C_2$, its combination with the other concepts would yield a better overall performance. For instance, two concepts may be redundant (e.g., "Linked Data" and "RDF"), therefore after one of them is selected, adding also the other would yield only a marginal advantage. In this section, we introduce a technique that addresses both limitations.

A comprehensive search outside the space of parameters is computationally intractable since it would need to test all possible permutations. For this reason, performing feature selection in large dimensional input spaces usually involves greedy algorithms. An approach to address this issue in the field of machine learning is the Recursive Feature Elimination algorithm [31], often used with Support Vector Machines and other classifiers. This approach iteratively constructs a model with a set of features, computes their weights, and removes the least important features, until the goal is reached. A crucial advantage of this method is that it takes into account the feature synergy and preserves features whose usefulness requires other features.

We thus adopted a similar procedure, that we label *Recursive Concept Elimination* (*RCE*), as the second step of POE. RCE generates an ontology composed of *n* concepts by applying the following steps:

1. It produces an ontology with $m$ concepts (where $m > n$) using the best set of parameters detected in the first phase (Sect. 4.4). If no ontology of $m$ concepts complies with the requirements, these are temporarily relaxed.
2. It ranks the concepts according to their importance for the tasks by generating $m - 1$ representations of the instances, each of them lacking a concept, and evaluating them. Each concept is given a weight equal to 1 minus the metric yielded by the evaluation of the representation from which it is absent [32].
3. It discards the $j$ concepts with the smaller weights and returns to step 2, until it reaches $n$ concepts. Finally, it returns the optimized ontology and the ranked set of parameters from the previous phase.

While it is technically possible to directly apply RCE to the full set of candidate concepts, it would not be computationally feasible in most cases. Using the set of best parameters to create an initial ontology of $m$ concepts allows us to obtain a tractable number of RCE iterations.

A further advantage of this method is that it allows users to understand exactly why a concept is there and in which way it relates with the dimensions discussed in Sect. 4.2 and with its performance on a task. Indeed, the ranking order will still be consistent with the set of parameters selected in the first phase and the absence of a concept from the original ranked list would be due to its insufficient performance with regard to the task. The user is thus able to review this information and test different solutions by modulating the input parameters.

## 5 Evaluation

We tested POE on the task of evolving the PMC taxonomy and used as instances a dataset of Springer Nature publications including 1,218 books in the 2012–2016 period. The evaluation had three aims. First, we wanted to compare POE versus alternative baselines from the state of the art, such as the TF-IDF method adopted in Text2Onto [14] and SPRAT [15]. Secondly, we intended to investigate whether optimizing for a certain task would also yield good performance on related ones. Finally, we intended to assess the effect of training POE on multiple tasks at once.

We thus compared the performance of the ontologies produced by different approaches in supporting the four tasks implemented in POE: automatic tagging (**Task 1**), similarity computation (**Task 2**), generation of recommendations (**Task 3**), and clustering (**Task 4**). In addition to the SN dataset, we adopted the Rexplore dataset [23] as the external source from which to derive statistics, such as the concepts frequencies described in Sect. 4.2 and TF-IDF. The Rexplore dataset is more generic than the SN one and contains 16 million research papers in the field of Computer Science from a variety of academic publishers.

We focused on the evolution of the branches under five concepts of the original PMC taxonomy: *I21017-Artificial Intelligence (incl. Robotics)*, *I23050-Computational Biology/Bioinformatics*, *I14050-Systems and Data Security*, *I14029-Software Engineering*, and *I13022-Computer Communication Networks*. These concepts were mapped to nine CSO concepts: Artificial Intelligence, Robotics, Bioinformatics, Cryptography, Access Control, Software Engineering, Software Design, Computer Networks, and Wireless Telecommunication Systems. Finally, their 2,451 sub-topics were selected as candidate concepts.

We tested fourteen alternative approaches:

- Term Frequency in the SN dataset (**FS**), ranking concepts according to their frequency in SN dataset.
- Term Frequency in the Rexplore dataset (**FR**).
- TF-IDF in the SN dataset (**TS**) (as in [14, 15]), considering the instances under the five branches for the TF and all the instances for the IDF.
- TF-IDF in the Rexplore dataset (**TR**).
- The parameter optimization in POE (Sect. 4.4), yielding the ontology produced from the best combination of parameters for instance tagging (**P1**), similarity computation (**P2**), generation of recommendations (**P3**), clustering (**P4**), and all these tasks together (**P5**).
- The full POE framework returning an ontology optimized for instance tagging (**POE1**), similarity computation (**POE2**), generation of recommendations (**POE3**), clustering (**POE4**), and all these tasks together (**POE5**).

We simulated a realistic situation by training the approaches and computing all the statistics (e.g., TF-IDF) in the 2012–2014 period and then evaluating their performance in the 2015–2016 period. In order to do so, we split the instances dataset in a training set of 718 books and a testing set of 500 books.

We then generated, for each approach, four evolved versions of PMC that included 20, 40, 60, and 80 new concepts and compared their performance using the metrics described in Sect. 4.3. The minimum and maximum number of concepts allowed for each of the five branches was set respectively to 4 and 25. RCE was performed by setting $m = n + 20$ and eliminating one concept at each iteration. POE was implemented in Python and ran on a 2.40 GHz Intel Xeon processor taking between 1 (POE1) and 8 (POE5) hours depending on the task. The computing time is usually not an issue for this kind of task, but if needed it could be cut down by parallelising the parameter optimization and the RCE phase. The existence of statistical differences between the two approaches was explored with the non-parametric Wilcoxon's signed rank test for matched variables.

The material produced during the evaluation and further details about the settings of the approaches are available at http://rexplore.kmi.open.ac.uk/POE.

Tables 1 and 2 show the performance of the approaches on the four tasks. The full version of POE optimized for a task (e.g., POE1 for task 1) obtained the best average result for the task in every case, outperforming both parameter optimization ($p = 0.002$ with Wilcoxon's rank test), and the other baselines ($p = 0.0004$). It also obtained the best result for each concept number, with the exception of few cases in which it was outranked by a different version of POE optimized for a similar task. POE5, the version optimized on all tasks at once, proved to be a good compromise by yielding on each task a performance marginally inferior or equal (in case of task 3) to the version of POE specifically optimized for the task ($p \geq 0.10$). In addition, the parameter optimization step optimized for a task (e.g., P1 for task 1) yielded better results than FS, FR, TS, TR on that same task ($p = 0.0004$).

**Table 1.** Performance in task 1 (instance tagging) on the left and task 2 (similarity computation) on the right. In bold the best results. In light grey the version of POE optimized for the task.

| | Task 1 (Instance Tagging) | | | | | Task 2 (Similarity Computation) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **20** | **40** | **60** | **80** | **aver.** | **20** | **40** | **60** | **80** | **aver.** |
| **FS** | 0.794 | 0.946 | 0.970 | 0.978 | 0.922 | 0.815 | 0.862 | 0.882 | 0.886 | 0.861 |
| **FR** | 0.818 | 0.896 | 0.924 | 0.972 | 0.903 | 0.820 | 0.852 | 0.862 | 0.898 | 0.858 |
| **TS** | 0.806 | 0.908 | 0.944 | 0.972 | 0.908 | 0.821 | 0.853 | 0.875 | 0.882 | 0.858 |
| **TR** | 0.808 | 0.916 | 0.918 | 0.938 | 0.895 | 0.821 | 0.859 | 0.867 | 0.886 | 0.859 |
| **P1** | 0.858 | 0.978 | **0.990** | 0.992 | 0.955 | 0.843 | 0.906 | 0.933 | 0.945 | 0.907 |
| **P2** | 0.858 | 0.978 | **0.990** | 0.992 | 0.955 | 0.843 | 0.906 | 0.934 | 0.945 | 0.907 |
| **P3** | 0.872 | 0.978 | 0.986 | 0.992 | 0.957 | 0.842 | 0.906 | 0.933 | 0.944 | 0.906 |
| **P4** | 0.856 | 0.956 | 0.978 | 0.992 | 0.946 | 0.842 | 0.883 | 0.931 | 0.945 | 0.900 |
| **P5** | 0.858 | 0.982 | **0.990** | 0.992 | 0.956 | 0.843 | 0.903 | 0.934 | 0.945 | 0.906 |
| **POE1** | **0.910** | **0.988** | 0.984 | **0.994** | **0.969** | 0.857 | 0.899 | 0.932 | 0.922 | 0.903 |
| **POE2** | 0.898 | 0.986 | 0.986 | 0.992 | 0.966 | 0.863 | **0.913** | **0.941** | **0.947** | **0.916** |
| **POE3** | 0.906 | 0.982 | **0.990** | 0.992 | 0.968 | 0.861 | **0.913** | 0.939 | 0.946 | 0.915 |
| **POE4** | 0.834 | 0.974 | 0.984 | 0.988 | 0.945 | 0.842 | 0.899 | 0.933 | 0.944 | 0.904 |
| **POE5** | 0.904 | **0.988** | 0.984 | 0.992 | 0.967 | **0.864** | 0.907 | 0.938 | 0.946 | 0.914 |

**Table 2.** Performance in task 3 (generation of recommendations) on the left and task 4 (clustering) on the right. In bold the best results. In light grey the version of POE optimized for the task.

| | Task 3 (Generation of Recommendations) | | | | | Task 4 (Clustering) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **20** | **40** | **60** | **80** | **aver.** | **20** | **40** | **60** | **80** | **aver.** |
| **FS** | 0.940 | 0.957 | 0.965 | 0.966 | 0.957 | 0.940 | 0.866 | 0.950 | 0.949 | 0.926 |
| **FR** | 0.943 | 0.953 | 0.957 | 0.974 | 0.957 | 0.943 | 0.868 | 0.850 | 0.962 | 0.906 |
| **TS** | 0.941 | 0.954 | 0.964 | 0.966 | 0.956 | 0.941 | 0.925 | 0.887 | 0.892 | 0.911 |
| **TR** | 0.941 | 0.957 | 0.959 | 0.971 | 0.957 | 0.941 | 0.894 | 0.934 | 0.954 | 0.931 |
| **P1** | 0.949 | 0.979 | 0.989 | 0.992 | 0.978 | 0.937 | 0.968 | 0.983 | **0.994** | 0.970 |
| **P2** | 0.949 | 0.979 | 0.990 | 0.992 | 0.978 | 0.937 | 0.968 | 0.983 | **0.994** | 0.970 |
| **P3** | 0.951 | 0.979 | 0.989 | 0.992 | 0.978 | 0.939 | 0.968 | 0.983 | 0.982 | 0.968 |
| **P4** | 0.950 | 0.970 | 0.988 | 0.992 | 0.975 | 0.944 | 0.963 | 0.977 | **0.994** | 0.969 |
| **P5** | 0.949 | 0.978 | 0.990 | 0.992 | 0.977 | 0.937 | 0.975 | 0.983 | **0.994** | 0.972 |
| **POE1** | 0.958 | 0.978 | 0.989 | 0.980 | 0.976 | 0.945 | 0.977 | 0.909 | 0.961 | 0.948 |
| **POE2** | 0.962 | 0.981 | 0.990 | 0.992 | 0.981 | 0.949 | 0.973 | 0.982 | **0.994** | 0.974 |
| **POE3** | 0.963 | **0.982** | **0.991** | **0.993** | **0.982** | 0.946 | **0.979** | 0.984 | 0.987 | 0.974 |
| **POE4** | 0.953 | 0.976 | 0.989 | **0.993** | 0.978 | **0.951** | 0.971 | **0.985** | **0.994** | **0.975** |
| **POE5** | **0.964** | 0.981 | **0.991** | 0.992 | **0.982** | 0.947 | 0.968 | 0.983 | 0.994 | 0.973 |

Furthermore, all the approaches optimized on one the four tasks (including POE5) performed significantly better ($p < 0.0001$) than the ones that simply used statistical techniques. Therefore, it seems that optimizing for one of these tasks holds benefits also on the other ones.

POE3, POE2, and POE5 yielded very good results on all tasks, obtaining the highest average performances, respectively 0.960, 0.959 and 0.959. Interestingly, the performance of POE2 and POE3 on task 4 (clustering) was only slightly inferior to POE4, while the performance of POE4 on task 2 and 3 was not as good. This is probably due to the fact that both task 2 and task 3 concern the similarity between instances, which is also used by K-Means++ for producing the cluster set.

## 6 Conclusions

We presented the Pragmatic Ontology Evolution (POE) framework, a novel approach that selects concepts to be included in an evolving ontology in accordance with user requirements and their impact on computational tasks. The evaluation showed that the full version of POE outperforms both parameter optimization ($p = 0.002$) and the other baselines ($p = 0.0004$).

While POE was initially conceived in the context of tackling a concrete real-world ontology evolution problem, the approach is generally applicable and opens up many interesting avenues of work. In particular, we intend to apply POE on different kinds of ontologies and computational tasks to derive some useful guidelines on how to balance users and application needs. We also intend to further enrich POE by allowing it to handle more complex candidate changes, involving different kinds of semantic relationships. Finally, on the technology transfer side, we will continue our collaboration with Springer Nature, with the aim of supporting its deployment within the editorial team, thus providing a powerful and user-friendly solution to facilitate the process of maintaining and evolving their editorial ontologies.

**Acknowledgements.** We would like to thank Springer DE for providing us with access to their large repositories of scholarly data.

## References

1. Ding, L., Kolari, P., Ding, Z., Avancha, S.: Using ontologies in the semantic web: a survey. In: Sharman, R., Kishore, R., Ramesh, R. (eds.) Ontologies. ISIS, vol. 14, pp. 79–113. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-37022-4_4
2. Osborne, F., Salatino, A., Birukou, A., Motta, E.: Automatic classification of Springer Nature proceedings with smart topic miner. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 383–399. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_33
3. Middleton, S.E., De Roure, D., Shadbolt, N.R.: Ontology-based recommender systems. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. INFOSYS, pp. 779–796. Springer, Berlin (2009). https://doi.org/10.1007/978-3-540-24750-0_24
4. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: Data Mining, ICDM 2003. IEEE (2003)
5. Osborne, F., Scavo, G., Motta, E.: Identifying diachronic topic-based research communities by clustering shared research trajectories. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 114–129. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_9

6. Osborne, F., Mannocci, A., Motta, E.: Forecasting the spreading of technologies in research communities. In: K-CAP 2017, Austin, Texas, USA (2017)
7. Zablith, F., et al.: Ontology evolution: a process-centric survey. Knowl. Eng. Rev. **30**(1), 45–75 (2015)
8. Stojanovic, L.: Methods and tools for ontology evolution (2004)
9. Klein, M., Noy, N.F.: A component-based framework for ontology evolution. In: Workshop on Ontologies and Distributed Systems at IJCAI, vol. 3, p. 4 (2003)
10. Noy, N.F., Chugh, A., Liu, W., Musen, M.A.: A framework for ontology evolution in collaborative environments. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 544–558. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_39
11. Zablith, F.: Evolva: a comprehensive approach to ontology evolution. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 944–948. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02121-3_87
12. Ristoski, P., Paulheim, H.: Semantic web in data mining and knowledge discovery: a comprehensive survey. Web Seman.: Sci. Serv. Agents World Wide Web **36**, 1–22 (2016)
13. Thanapalasingam, T., Osborne, F., Birukou, A., Motta, E.: Ontology-based recommendation of editorial products. In: International Semantic Web Conference 2018, Monterey, CA, USA (2018)
14. Cimiano, P., Völker, J.: Text2Onto. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005). https://doi.org/10.1007/11428817_21
15. Maynard, D., Funk, A., Peters, W.: SPRAT: a tool for automatic semantic pattern-based ontology population. In: International Conference for Digital Libraries and the Semantic Web, Trento, Italy (2009)
16. Novacek, V., Handschuh, S.: Semi-automatic integration of learned ontologies into a collaborative framework (2007)
17. Huang, Z., Stuckenschmidt, H.: Reasoning with multi-version ontologies: a temporal logic approach. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 398–412. Springer, Heidelberg (2005). https://doi.org/10.1007/11574620_30
18. Xuan, D.N., Bellatreche, L., Pierra, G.: A versioning management model for ontology-based data warehouses. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 195–206. Springer, Heidelberg (2006). https://doi.org/10.1007/11823728_19
19. Wang, Y., Liu, X., Ye, R.: Ontology evolution issues in adaptable information management systems. In: 2008 IEEE International Conference on e-Business Engineering, ICEBE 2008, pp. 753–758. IEEE (2008)
20. Liang, Y., Alani, H., Shadbolt, N.: Changing ontology breaks queries. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 982–985. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_79
21. Kondylakis, H., Plexousakis, D.: Ontology evolution: assisting query migration. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 331–344. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34002-4_26
22. Osborne, F., Motta, E.: Klink-2: integrating multiple web sources to generate semantic topic networks. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 408–424. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_24
23. Osborne, F., Motta, E., Mulholland, P.: Exploring scholarly data with rexplore. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 460–477. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_29
24. Salatino, A.A., Thanapalasingam, T., Mannocci, A., Osborne, F., Motta, E.: The computer science ontology: a large-scale taxonomy of research areas. In: International Semantic Web Conference 2018, Monterey, CA, USA (2018)

25. Klein, M.C., Fensel, D.: Ontology versioning on the Semantic Web. In: SWWS, pp. 75–91 (2001)
26. Sellami, Z., Camps, V., Aussenac-Gilles, N.: DYNAMO-MAS: a multi-agent system for ontology evolution from text. J. Data Semant. **2**(2–3), 145–161 (2013)
27. Sabou, M., Fernandez, M., Motta, E.: Evaluating semantic relations by exploring ontologies on the semantic web. In: Horacek, H., Métais, E., Muñoz, R., Wolska, M. (eds.) NLDB 2009. LNCS, vol. 5723, pp. 269–280. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12550-8_22
28. Qin, L., Atluri, V.: Evaluating the validity of data instances against ontology evolution over the semantic web. Inf. Softw. Technol. **51**(1), 83–97 (2009)
29. Hartung, M., Kirsten, T., Rahm, E.: Analyzing the evolution of life science ontologies and mappings. In: Bairoch, A., Cohen-Boulakia, S., Froidevaux, C. (eds.) DILS 2008. LNCS, vol. 5109, pp. 11–27. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69828-9_4
30. Groß, A., Hartung, M., Prüfer, K., Kelso, J., Rahm, E.: Impact of ontology evolution on functional analyses. Bioinformatics **28**(20), 2671–2677 (2012)
31. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1–3), 389–422 (2002)
32. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)