Towards Feature Selection for Ranking and Classification Exploiting Quantum Annealers

Maurizio Ferrari Dacrema
Politecnico di Milano
Italy
maurizio.ferrari@polimi.it

Fabio Moroni
Politecnico di Milano
Italy
fabio6.moroni@mail.polimi.it

Riccardo Nembrini
Politecnico di Milano, ContentWise
Italy
riccardo.nembrini@polimi.it

Nicola Ferro
Università degli Studi di Padova
Italy
ferro@dei.unipd.it

Guglielmo Faggioli
Università degli Studi di Padova
Italy
faggioli@dei.unipd.it

Paolo Cremonesi
Politecnico di Milano
Italy
paolo.cremonesi@polimi.it

ABSTRACT

Feature selection is a common step in many ranking, classification, or prediction tasks and serves many purposes. By removing redundant or noisy features, the accuracy of ranking or classification can be improved and the computational cost of the subsequent learning steps can be reduced. However, feature selection can be itself a computationally expensive process. While for decades confined to theoretical algorithmic papers, quantum computing is now becoming a viable tool to tackle realistic problems, in particular special-purpose solvers based on the Quantum Annealing paradigm. This paper aims to explore the feasibility of using currently available quantum computing architectures to solve some quadratic feature selection algorithms for both ranking and classification.

The experimental analysis includes 15 state-of-the-art datasets. The effectiveness obtained with quantum computing hardware is comparable to that of classical solvers, indicating that quantum computers are now reliable enough to tackle interesting problems. In terms of scalability, current generation quantum computers are able to provide a limited speedup over certain classical algorithms and hybrid quantum-classical strategies show lower computational cost for problems of more than a thousand features.

CCS CONCEPTS

• **Information systems** → **Content analysis and feature selection**; • **Computer systems organization** → **Quantum computing**.

KEYWORDS

Feature Selection, Quantum Computing, Quantum Annealing, Machine Learning

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531755

ACM Reference Format:
Maurizio Ferrari Dacrema, Fabio Moroni, Riccardo Nembrini, Nicola Ferro, Guglielmo Faggioli, and Paolo Cremonesi. 2022. Towards Feature Selection for Ranking and Classification Exploiting Quantum Annealers. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531755

1 INTRODUCTION

Information Retrieval (IR) is concerned with delivering relevant information to people, according to their information needs, context, and profile, in the most effective and efficient way possible. Central to this goal are *ranking* and *classification*, often exploited in conjunction. Machine learning approaches have been widely investigated for this purpose. These methods however suffer from the known feature selection problem. As the data becomes more rich and complex, identifying the relevant features may require to evaluate an exponentially increasing number of cases which rapidly becomes prohibitively resource intensive. The feature selection problem is mitigated by deep learning and, more generally, neural approaches that have gained popularity in recent years. Despite these methods being extremely versatile and generally able to provide good overall effectiveness, it is known their performance is not always stable and may vary a lot across topics, for example the performance may improve for half of the topics while degrade for the other half [30]. A further disadvantage is that these neural approaches are very demanding in terms of computing resources and require enormous amounts of data which leads to larger and larger models that are not free from risks, as pointed out by Bender et al. [6].

In this paper we take a step back and wonder ourselves if it is possible to make the feature selection problem more "affordable" in order to make more appealing the use of "traditional" machine learning approaches for ranking and classification. To this end, we investigate the feasibility of and how to apply current generation quantum computing technologies to improve feature selection. To the best of our knowledge very little work has been done to asses the effectiveness and efficiency of such technologies to tackle feature selection problems, especially for both ranking and classification. For example, Nembrini et al. [33] proposed a heuristic to build a hybrid recommender system that selects important features according

to how well they allow us to approximate another recommendation model based on the user behavior.

The contributions of this paper are the following:

- formulate the feature selection problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem which can be solved using Quantum Annealing (QA);
- compare efficiency and effectiveness of classical and quantum QUBO problems solvers;
- show that a quantum computer is able to more efficiently solve the feature selection problem, for both ranking and classification, with an effectiveness comparable to classical solvers;
- show that the quantum-classical hybrid approaches exhibit better scalability when the number of features increases compared to the classical ones.

The results reported in this paper show that quantum computing approaches have become a viable option for the feature selection problem in IR and that they are worthy of further investigation.

The paper is organized as follows: Section 2 describes related work on quantum computing and feature selection, Section 3 presents feature selection as a Quadratic Unconstrained Binary Optimization problem as well as classical and quantum strategies to solve such problems, Section 4 describes the experimental pipeline, Section 5 presents the results, finally Section 6 draws the conclusions and presents future research directions.

2 RELATED WORKS

Quantum Computing. In recent years, quantum computing is gaining popularity as a new computational paradigm able to offer speedups for several computational tasks that are difficult on classical hardware. Quantum computing is often used to refer to a computational paradigm called *gate model* or *universal quantum computer*, in which a quantum state is manipulated with a controlled sequence of operations performed via quantum gates, in a way that is not dissimilar from how classical computers operate. Universal quantum computers are theoretically able to compute any function and have rigorously proven speedups for certain tasks. On the other hand, they are currently quite unreliable due to noise and are available only with a limited number of qubits, the basic unit of quantum information. Furthermore, this paradigm requires specially designed algorithms that is challenging to develop.

Another paradigm of quantum computing is Quantum Annealing (QA) in which a special purpose device, called Quantum Annealer, is used to rapidly sample optimal solutions of a QUBO problem; note that the detailed description of QUBO methods is deferred to subsection 3.1. As opposed to universal quantum computers, Quantum Annealers have a limited computational flexibility, being able only to tackle optimization problems, and the question of proving the type of speedup they offer is still open. On the other hand, they suffer much less from noise and are available with a rather large number of qubits, allowing to tackle problems of interesting size. Moreover, the simplicity of the required QUBO formulation and fast solution time have made it a promising and widely accessible technology.

This has fueled significant research from both industry and academia to explore the potential of this technology. Several QUBO formulations for important problems have been developed such as graph partitioning [5, 45], Support Vector Machines [48], Restricted Boltzmann Machines [1, 3] and optimization for resource allocation [11]. QA has also been applied to collaborative filtering [17] and the personalization of the user interface in a recommender system [14].

For these reasons, this paper investigates feature selection via QA and QUBO formulation.

Feature Selection. Feature selection is one of the problems that fit well with the mathematical formulation required to use a Quantum Annealer. Its goal is to isolate a subset of all available features in order to improve the effectiveness of a model, being it for ranking or classification, and/or to reduce the computational cost. Feature selection is a widely researched topic [12, 24, 40] and the main approaches can be divided in three categories:

Filter methods: the features are selected based on information-theoretical measures that do not optimize the model itself, e.g., variance or entropy.

Embedded methods: the selection, or weighting, of the features is a part of the model training, e.g., lasso or ridge regression, neural networks and factorization machines all learn a weight for the input feature data.

Wrapper methods: the model is considered as a black box and the features are selected as those that optimize the end effectiveness of the model.

This work focuses on filter methods, since they are agnostic about the model to optimize and are well suited to study and compare several QA and QUBO methods across a variety of models.

Filter methods for features selection can be further categorized into linear and quadratic approaches. The former consider individual features singularly while the latter also account for feature interaction. Among the most common linear approaches, we can list the following:

ANOVA F-Test: the Analysis of Variance [7, 16, 41] method attempts to minimize false negative errors.

Chi2 Test: features are ranked according to their Chi-square statistic value. This test aims to identify the features that are more likely independent from the target label.

Mutual Information: features are ranked according to their Mutual Information with respect to the target variable. The Mutual Information is a statistical measure also known as *Shannon's Information* [36, 43, 49].

Pearson Correlation: features are ranked according to their *Pearson Correlation* with the target variable.

Linear Boosting: each feature is used to train a Support Vector Classifier, which is then used to label the samples. Features are ranked according to the *Pearson Correlation* between the prediction of the classifier and the target variable.

Variance Threshold: removes low-variance features, the feature score is computed as the variance of its values.

These approaches will be considered as plain baselines in the subsequent experiments. In Section 3.1 we will describe quadratic approaches in detail, since they are the focus of our QUBO optimization.

Quantum Information Retrieval. Quantum IR is a branch of IR initiated by van Rijsbergen [46] which exploits the formalism and concepts of quantum mechanics, such as Hilbert spaces and Hermitian operators, to formulate and describe IR models and specific problems, such has relevance feedback. Over the years Quantum IR has evolved [32], exploring mainly two areas – (i) representation and ranking; (ii) user interaction – leveraging "Hilbert space models for representation learning and quantum probability rules to model cognitive interference in document ranking" [44].

This work does not deal with Quantum IR but rather explores how Quantum Computing, in particular Quantum Annealing, can be applied to directly solve the feature selection problem in ranking and classification for IR.

Efficient Learning to Rank (LtR). A further line of work strongly connected to this work concerns the study of efficient approaches for feature selection applied to the LtR task. Among the most prominent efforts in this field, we list the one from Gigli et al. [20]. In [20], the authors propose to use the Hierarchical agglomerative Clustering Algorithm for feature Selection (HCAS) to select features to be fed to a LtR algorithm in both a fast and effective way. Following a similar line of thoughts, Lucchese and Nardini [29] highlight the importance of feature selection to obtain computationally efficient LtR solutions. In particular, in [29], the authors reiterate the extent of the trade-off between efficiency and effectiveness that characterize LtR and the role of features selection in improving the former without excessively degrading the latter. Finally, Lai et al. [27] propose an optimization strategy for feature selection while Purpura et al. [37] exploit a neural networks based approach. Both endeavours mentioned above do not focus on efficiency aspects but highlight the improved effectiveness of LtR approaches if used in combination with feature selection solutions.

3 METHODOLOGY

3.1 Feature Selection as a Quadratic Problem

Identifying the appropriate features is not trivial and depends on the data, task and context of interest, e.g., obtain the best possible model accuracy, discard correlated features to reduce redundancy, minimize the number of selected features to improve scalability, and so on.

Quadratic models for feature weighting and selection have been studied for several years [25, 40]. While *feature weighting* is a simpler task, since when the problem coefficients are semi-definite positive the optimization problem is convex and can be solved rapidly, *feature selection* is an NP-hard problem. In this work, we focus only on feature selection, since it is the hardest task and can possibly lead to bigger gains. This section describes three quadratic models for feature selection represented with the QUBO formulation of optimization problems, which allows us to easily describe many important NP-complete and NP-hard problems [21, 28]. The QUBO formulation is defined as follows:

$$\min \quad y = x^T Q x$$

where $x \in \{0,1\}^m$ is a vector of m binary variables and Q is an $m \times m$ matrix that defines the function to optimize. In all the following methods, each feature will be associated to a binary variable to indicate whether it should be selected or not, therefore $m = |F|$ with F the set of existing features.

3.1.1 MIQUBO. Mutual Information QUBO (MIQUBO) is a quadratic feature selection model based on Mutual Information. Its objective is to maximize the mutual information between selected features and the target variable as well as the conditional mutual information between a feature and the target, given other selected features. MIQUBO is defined as:

$$Q_{ij} = \begin{cases} -MI(f_i, y|f_j) & if \;\; i \neq j \\ -MI(f_i, y) & if \;\; i = j \end{cases}$$

where $MI(f_i, y)$ is the mutual information [36, 43, 49] between feature f_i and target y and $MI(f_i, y|f_j)$ is the conditional mutual information between feature f_i and target y given feature f_j. Since mutual information is always non-negative, MIQUBO has a trivial solution where all features are selected. Due to this MIQUBO requires to specify what is the desired number of feature to select by introducing a penalization term.

3.1.2 QUBO-Correlation. This strategy, originally proposed by Ferreira and Figueiredo [18], is based on the Pearson Correlation. QUBO-Correlation aims to maximize the correlation between the selected features and the target variable, but also to minimize the correlation among the selected features in order to reduce redundancy. QUBO-Correlation is defined as:

$$Q_{ij} = \begin{cases} -r(f_i, f_j) & if \;\; i \neq j \\ r(f_i, y) & if \;\; i = j \end{cases}$$

where $r(\cdot, \cdot)$ is the Pearson's r Correlation.

3.1.3 QUBO-Boosting. This technique estimates the information content of a feature based on the predictions computed by a classifier using only that feature and is inspired by the work done by Neven et al. [34]. QUBO-Boosting operates by first training $|F|$ different Support Vector Classifier (SVC), one per each existing feature. The trained classifier is then used to compute the predicted class for each sample. The correlation between the predicted label and the correct one is used to assess the information content of the feature. QUBO-Boosting is defined as follows:

$$Q_{ij} = \begin{cases} r(h_i, h_j) & if \;\; i \neq j \\ \frac{S}{|F|^2} + \lambda - 2 * r(h_i, y) & if \;\; i = j \end{cases}$$

where h_i is the predicted classification of the SVC trained only on feature i, S is the number of samples in the dataset and λ is a hyperparameter. Note that the classifier used to compute the coefficients is independent from the one that will be trained using the selected features and both can be changed freely depending on the scenario of interest.

3.1.4 Selecting k features. In order to control the selection of a desired number of features, it is possible to include in the objective function a penalty term. This penalty term will be minimized when k variables have value 1. The resulting optimization problem becomes:

$$\min \quad y = x^T Q x + \left(\sum_{i=1}^{N} x_i - k \right)^2$$

Note that this penalty term is essential for methods like MIQUBO that have a trivial solution where all features are selected, while for the other methods it allows to improve the control on the final outcome.

3.2 Solving QUBO with Traditional Approaches

The QUBO formulation is rather flexible and such problems can be solved with several techniques. Among them, in this work we consider the following:

- **Simulated Annealing (SA):** it is a metaheuristic used to search for solutions of discrete optimization problems [26]. Based on a temperature parameter decreasing to zero, it searches locally for better solutions while accepting energy-increasing moves with a certain probability that depends on the temperature.
- **Tabu Search (TS):** it is a metahueristic for discrete optimization problems [35]. It performs local search and accepts energy increasing moves only if no improving move is available. Moreover, it forbids moves to already visited states.
- **Steepest Descent (SD):** starting from an initial solution, at each step this algorithm performs the variable flip that reduces the energy the most.

3.3 Solving QUBO with Quantum Annealing

Besides traditional approaches, QUBO problems are particularly well suited to be solved via Quantum Annealing (QA). The term QA [4] originally referred to a meta-heuristic that was proposed as an improvement of Simulated Annealing (SA) [2] for the optimization of objective functions where the quality of a solution is seen as analogous to an *energy*. In particular, SA operates by simulating thermal fluctuations and is prone to remain trapped into local optima that are surrounded by a high energy barrier, i.e., the worse are the solutions adjacent to the current one, the more difficult for SA is to explore the solution space. QA was instead designed to leverage quantum tunneling in such a way that even when surrounded by a high energy barrier there is a certain probability to *tunnel* through and escape the local optima. Quantum tunneling is a well-understood quantum mechanical phenomena and, as such, the original proposal for QA simulated the process on the available classical systems.

This work instead adopts a different approach to QA, that is to leverage a special-purpose physical device that already exhibits the needed quantum behavior, i.e., a Quantum Annealer or Quantum Processing Unit (QPU). In a QPU the function to optimize is represented as the energy landscape of a physical system, referred to as the *Hamiltonian* [15], and the device acts as a sampler of low-energy solutions.

3.3.1 Steps to Use a QPU. Using a special-purpose device to solve QUBO problems brings advantages and disadvantages. The most significant advantage is the very low time-to-solution, as hundreds of solutions can be sampled in few milliseconds. As a downside, a physical device has stringent constraints on the size of the problems it can be used to solve. In particular, in a QPU each variable is represented with a qubit. The qubits are connected according to a certain topology and only a limited number of connections between qubits exist. Connections corresponds to entries in the Q matrix of the QUBO problem. In this paper the D-Wave Advantage QPU is used, which has 5600 qubit and a topology called Pegasus, in which every qubit is connected to 15 others [8]. This quantum computer easily accessible on the cloud.[1]

Problem Formulation. A QPU operates by finding the minimums of a certain energy landscape, called Hamiltonian. The Hamiltonian is described as a function of the QUBO formulation. The QPU can use the QUBO formulation directly and therefore there is no need for more complex representations as is instead the case for universal quantum computers.

Embedding a QUBO problem on the QPU. When using a QPU to solve a QUBO problem, the first step is to ensure that the problem can fit on the hardware given its limited connectivity and number of qubits. This step is called *minor embedding* [13] and operates by transforming the original QUBO problem in an equivalent one that accounts for the QPU limited connectivity. This is done, for example, by creating additional variables that will inherit some of the connections of the original ones. Figure 1 shows an example of a QUBO problem with a triangular structure, that is three variables connected to each other. If such a structure cannot be mapped in the device, the problem structure is transformed into a square by creating an auxiliary variable, therefore using two qubits to represent a single variable. Due to this process, it is not easy to define a relation between the number of QUBO problem variables and the number of qubits required by the QPU, as the topology of the QUBO problem (i.e., the structure of the Q matrix) plays a significant role. This embedding phase can be performed with algorithms that run in polynomial time [13].

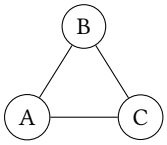
Sampling Solutions. Once the problem has been embedded, it is possible to submit the problem to the QPU via APIs. The device operates by *sampling* low-energy solutions by repeating the physical QA process multiple times for the desired length of time. Depending on the problem, it may be more ore less likely that a good solution is sampled, therefore it is common to sample a significant number of solutions, typical values are $10^2 - 10^4$. Each solution is associated to the corresponding energy, which can be used to select the desired ones.

Advanced Controls. The QPU has several settings that can be used to control the underlying physical process, such as the duration of the annealing process that returns a single sample (typical durations are between $1 - 100\mu s$), the schedule of the annealing process, time offsets for different qubits etc. Finding the optimal setup of the physical process is a complex task that is yet not well-understood and therefore constitutes an open research question that goes beyond the scope of this work.

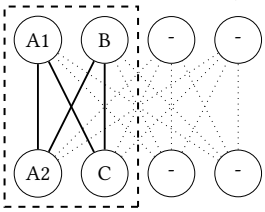
In this work, we explore two state-of-the-art approaches to QA:

- **Advantage Quantum Annealer (QPU):** uses a real QPU to search for solutions, with default hyperparameters (in particular, annealing time of $20\mu s$).
- **D-wave Leap Hybrid (Hybrid):** uses a hybrid quantum-classical approach to decompose the QUBO problem and partially

[1] https://www.dwavesys.com/solutions-and-products/cloud-platform/



(a) A QUBO problem with a triangular structure.



(b) A possible embedding into a Chimera graph.

Figure 1: Example of how a problem is embedded in a Quantum Annealer topology. Figure 1a shows the structure of a QUBO problem and Figure 1b shows a portion of the Chimera topology, an earlier version of Pegasus. Each node represents a qubit and each edge a connection. Each qubit is connected to 4 others of the same cell and to others in different cells. In this case, the triangular problem structure cannot fit directly into the Chimera cell topology therefore the problem variable *A* becomes a logical variable represented by two different qubits *A1* and *A2*.

solve smaller problems directly on the annealer; available as a cloud service from D-Wave Leap[2].

4 EXPERIMENTAL PIPELINE

The effectiveness of the QPU is assessed on two different tasks, classification and ranking. This section presents the tasks, the used datasets and algorithms as well as the details of the experimental pipeline. The source code to reproduce these experiments is publicly available online. [3]

4.1 Classification Task

The classification task consists in classifying a sample from a dataset (such as an image, a sound, a sentence) into two or more different classes, based on the sample's features. These features can be heterogeneous (numerical, categorical) and may refer to a direct measurement (e.g., the size of an object) or may be extracted from other data (e.g., the number of times a certain word appears in a text). Given the usually high number of features, feature selection is important and largely used in classification, in order to identify the most useful subset of features for the specific task.

Dataset. For the classification experiments, datasets are taken from OpenML [47], with a number of features ranging from 34 to 5000. The datasets come from heterogeneous fields and tasks, among which: clinical diagnosis (`waveform-5000`, `SPECTF`), organic chemistry (`Bioresponse`), forest cover type (`covertype`), duplicate locations resolution (`nomao`), digit recognition (`USPS`, `SVHN_small`, `gisette`), e-mail filtering (`spambase`). Some datasets have features extracted from images (`SPECTF`, `SVHN_small`, `USPS`, `gisette`), signals (`waveform-5000`), e-mail text (`spambase`) and even sound (`isolet`). The number of samples spans from a couple hundreds to tens of thousand, with one large dataset of hundreds of thousands samples (`covertype`). Almost every dataset has two classes, except from `waveform-5000` (3), `USPS` and `SVHN_small` (10) and `isolet` (26).

Classification algorithm. The algorithm used for the classification task is Random Forest, which is a widely used ensemble method that builds multiple decision trees. The class is determined by voting as the one selected by the most random trees. Random Forest has been chosen because it is able to provide good classification accuracy with limited fine-tuning and generally works well across multiple tasks and domains, which makes it well-suited for this study. The optimized metric is classification accuracy.

Data split. The dataset is randomly split in two sets, training and testing, respectively with 70% and 30% of the samples. The splits are generated by randomly sampling data samples but ensuring the approximate class distribution is maintained. This is especially important for datasets that exhibit class imbalance. Due to the limited size of some datasets, 5-fold Cross Validation is applied on the training data instead of creating a further split for validation.

4.2 Ranking Task

To evaluate the different feature selection strategies in a traditional IR setting, we consider the LtR task. The LtR task requires sorting a set of documents according to their expected relevance for a given query. For each query document pair, a set of features is computed. Additionally, for the pairs query document used during the training phase, we have the relevance judgments describing whether the document is relevant to the query. An LtR approach often consists in training a model, using relevance judgments as supervision, that uses features of query document pairs to predict the document relevance to the query. This, in turn, allows sorting the documents according to the predicted relevance for a query.

Dataset. Following previous literature on LtR, we adopt LETOR to assess the performance of different feature selectors. LETOR is a collection of datasets explicitly meant for the LtR task. Currently, there are two main versions, LETOR 3.0 [38] and LETOR 4.0 [39] that incorporate respectively seven and two distinct datasets.

LETOR 3.0 includes seven different datasets, among which in our experiments we consider `OHSUMED`. The corpus of this dataset contains more than 300,000 scientific publications (title and abstract of medical papers). `OHSUMED` includes 45 features. Relevance in the `OHSUMED` collection is ternary and can be either 0, 1, or 2. Such values are assigned respectively to non-relevant, partially relevant, and highly relevant documents.

LETOR 4.0 contains the `MQ2007` and `MQ2008` datasets. They are built upon TREC Million Queries track from the years 2007 and 2008 and have 1700 and 800 queries each. `MQ2007` and `MQ2008` are based on the `.gov2` collection that consists of more than 25 million documents. Akin to `OHSUMED`, relevance judgements for both `MQ2007` and `MQ2008` are ternary. For each pair query document to be reranked, LETOR 4.0 contains 46 features.

[2]D-Wave Leap - https://cloud.dwavesys.com/leap/
[3]https://github.com/qcpolimi/SIGIR22_QuantumFeatureSelection.git

Ranking algorithm. LambdaMART [9] is a state-of-the-art LtR approach that combines the LambdaRank [10] optimization algorithm and Multiple Additive Regression Trees (MART) [19]. We adopt the implementation of LambdaMART available in RankLib[4], with default hyperparameters. We used nDCG with a cutoff at ten as optimization measure, as commonly done in the literature.

Data split. Given the cost of selecting the features using quantum-based strategies, we limit our analysis to a single collection split. In particular, for each dataset, we use 60% of the queries as a training set – both select the features and train the classifiers. The remaining 40% of the queries in each dataset are split evenly in validation and test sets: the former allows selecting the optimal number of features, and the latter serves to evaluate the performance of the classifiers.

4.3 Selecting the optimal set of features

Feature selection is performed by applying to the training data QUBO and the baseline linear feature selection methods described in Section 2. Note that all methods require to specify the number of features to be selected, k, which constitutes a hyperparameter. How the k features are selected differs between linear filter methods and QUBO-based methods. For linear filter methods the k features selected are the ones with the highest computed scores. For QUBO methods a component is added to the loss function as described in Section 3.1.4. The search range for k depends on the number of features $|F|$ of the dataset. If there are less than 50 features, k is selected from a range between 1 and $|F|$ with a unitary step. Otherwise, a maximum of 50 values for k are explored, equally distributed between 1 and $|F|$ excluded.

For each of the explored k values, the resulting selected features are used to train either the classifier or the ranker, depending on the task, and the resulting model is evaluated on the validation data. Note that the number N of selected features may be different from k, the target number of features to be selected, because the target value k is expressed as a soft constraint. The N value associated to the features with the best model quality on the validation data is selected and a final model is trained on the union of training and validation data. The quality of this model evaluated on the test data is then reported.

For all QUBO solvers 100 solutions are generated or sampled, except for the Leap Hybrid method that only returns the best solution found according to its energy.

5 RESULTS

This section presents the results aiming to assess the capability of the QPU to solve QUBO problems competitively, by comparing the solutions obtained by all QUBO solvers both in terms of effectiveness and efficiency. Overall, more than 10k experiments were conducted, of which 900 on the QPU directly and 1650 using the Hybrid solver.

5.1 Effectiveness

The results obtained by the QUBO solvers are compared and displayed in Table 1 (Classification Task) and Table 2 (Ranking Task).

[4]https://sourceforge.net/p/lemur/wiki/RankLib/

To determine statistically significant differences between features selection strategies applied to the classification task, we employ the McNemar's test with significance level $\alpha = 0.05$ and Bonferroni correction following the procedure described by Japkowicz and Shah [23]. As a general trend, quantum solvers perform as well as traditional ones, as expected. The only exception to this pattern is represented by the `covertype` dataset, where, in several cases, there are statistically significant differences between different approaches. It should be noted that, even for `covertype`, there is no dominance of traditional solvers over quantum ones, nor vice versa: the best performing solver is linked to the chosen heuristics.

Concerning the ranking task, we assess the presence of statistically significant differences between selection strategies using a two-way ANOVA, with topic and feature selector factors, followed by Tukey's posthoc pairwise comparison procedure with significance level $\alpha = 0.05$. Such a comparison strategy follows the one proposed originally by [42], where the system factor is replaced with the feature selector one. On all collections, the statistical procedure does not deem any selector to be statistically better than others. This indicates overall comparable performance when using either traditional or quantum approaches to the solution of the QUBO problem: using quantum strategies, we can expect results that are at least as good as those that we would have achieved otherwise.

The first observation that can be made is that there is no single QUBO solver that is superior to the others, rather, the solver that is able to achieve the best result is different depending on the dataset. Across all experiments TS is able to reach the best result 11 times, SA and QPU 8 times, Hybrid 7 times and SD 6 times. This is likely due to the peculiarities of each dataset, task and, to some extent, the stochastic nature of some solvers. Some differences instead emerge by comparing across tasks, in particular no feature selection approach is able to improve the effectiveness on dataset `MQ2008` and the Hybrid solver is slightly less effective when applied to the Ranking task. The behavior of the various QUBO solvers remains instead consistent across the feature selection methods, although different QUBO heuristics result in different overall effectiveness. For example, MIQUBO appears to produce better results compared to the others.

Looking at the QPU solver, its effectiveness is very close, if not almost identical, to that of the other solvers, sometimes resulting in the best selection of features. In very few cases the solution obtained with the QPU is worse compared to the other solvers, but within 5% of the best one. This result indicates that the QPU is indeed a reliable solver that can be used to tackle real problems across different datasets, heuristics and tasks. Note however that the largest problem that could be solved directly on the QPU had 124 features. Although the QPU has more than 5000 qubits, the QUBO matrix resulting from the feature-selection problem is fully-connected and therefore its structure is difficult to fit on the limited connectivity structure of the QPU, therefore the problem size that can fit the hardware is greatly reduced. For larger problems it is still possible to use the Hybrid QPU-Classical approach, which was used for datasets of up to 5000 features but is able to tackle even larger problems.

In terms of the number of selected features, again no clear pattern emerges with no solver able to consistently provide better solutions

Table 1: Classification accuracy for all QUBO feature selection methods. The first column refers to all dataset features, the other columns refer to the various solvers used to solve the QUBO feature selection problem. Superscripts ^{A}, ^{D} and ^{T} indicate statistical difference between the Quantum Solvers and Simulated Annealing, Steepest Descent and Tabu Search respectively, determined using the procedure described in Section 5. The best results for each QUBO method and dataset are highlighted in bold. Results are missing for the QPU when the problem required more qubits than the available ones, in such instances the only Quantum Solver that could be used is Hybrid.

Method	Dataset	All Features		Quantum Solver				Traditional Solver					
				QPU		Hybrid		SA		SD		TS	
		F	Accuracy	N	Accuracy	N	Accuracy	N	Accuracy	N	Accuracy	N	Accuracy
QUBO Correlation	waveform-5000	40	0.8540	31	0.8393	36	0.8333	36	0.8320	34	0.8333	40	**0.8593**
	SPECTF	44	**0.8667**	14	0.8476	16	0.8286	18	0.8286	16	0.8381	15	0.8476
	covertype	54	0.9605	46	0.9526^{ADT}	43	0.9352^{T}	41	0.9347	43	0.9342	49	**0.9646**
	spambase	57	0.9631	47	0.9573	47	0.9594	51	0.9602	50	0.9587	56	**0.9660**
	nomao	118	0.9654	93	**0.9659**	104	0.9649	110	0.9654	111	0.9650	118	0.9654
	tecator	124	0.9167	67	**0.9306**	26	**0.9306**	26	0.9028	35	0.9167	79	0.9167
	USPS	256	**0.9642**		-	198	0.9599	201	0.9613	204	0.9624	253	0.9599
	isolet	617	**0.9432**		-	565	0.9402	565	0.9406	556	0.9402	617	0.9406
	Bioresponse	1776	0.7913		-	76	0.7940	76	0.7948	403	**0.8002**	387	0.7984
	SVHN_small	3072	**0.5801**		-	401	0.5170^{T}	401	0.5203	352	0.5166	1468	0.5619
	gisette	5000	0.9676		-	4536	0.9471^{T}	4536	0.9486	4536	0.9462	2560	**0.9681**
QUBO Boosting	waveform-5000	40	0.8540	23	0.8287	24	**0.8620**	23	0.8567	25	0.8520	28	0.8567
	SPECTF	44	0.8667	6	0.8286	30	**0.8857**	29	0.8571	26	0.8286	13	0.8571
	covertype	54	0.9605	33	0.9621^{ADT}	31	0.9667^{ADT}	30	**0.9697**	33	0.9643	25	0.9650
	spambase	57	**0.9631**	37	0.9558	35	0.9529	37	0.9573	36	0.9566	53	0.9587
	nomao	118	0.9654	64	0.9655	79	0.9645	79	0.9641	76	0.9650	112	**0.9667**
	tecator	124	0.9167	69	0.9167	23	0.9167	14	0.9167	40	0.8889	16	**0.9444**
	USPS	256	**0.9642**		-	161	0.9573	161	0.9613	165	0.9591	234	0.9624
	isolet	617	**0.9432**		-	467	0.9380	495	0.9415	495	0.9406	604	0.9380
	Bioresponse	1776	0.7913		-	1229	0.7922	997	0.7940	1345	0.7948	1123	**0.7993**
	SVHN_small	3072	0.5801		-	2245	0.5670	1855	0.5623	2099	**0.5831**	1469	0.5619
	gisette	5000	0.9676		-	1750	0.9729	2079	**0.9752**	1002	0.9733	2558	0.9690
MIQUBO	waveform-5000	40	0.8540	20	0.8547	20	0.8553	27	**0.8580**	38	0.8540	40	0.8493
	SPECTF	44	0.8667	44	**0.8857**	44	**0.8857**	44	**0.8857**	44	0.8476	44	0.8476
	covertype	54	0.9605	50	$\mathbf{0.9646}^{AT}$	49	0.9616^{DT}	36	0.9624	27	0.9642	54	0.9605
	spambase	57	0.9631	52	0.9638	53	0.9616	52	0.9645	54	0.9638	57	**0.9660**
	nomao	118	0.9654	96	**0.9667**	109	0.9656	118	0.9656	113	0.9661	118	0.9656
	tecator	124	**0.9167**	67	**0.9167**	8	**0.9167**	42	**0.9167**	19	**0.9167**	124	**0.9167**
	USPS	256	0.9642		-	219	0.9620	256	0.9624	185	**0.9649**	256	0.9634
	isolet	617	0.9432		-	501	**0.9436**	617	0.9389	600	0.9410	617	0.9427
	Bioresponse	1776	0.7913		-	1709	**0.7966**	1419	0.7922	1492	0.7869	1776	0.7922
	SVHN_small	3072	0.5801		-	1871	0.5787	1337	0.5787	2136	**0.5911**	1609	0.5697
	gisette	5000	0.9676		-	1076	0.9743	1382	**0.9771**	2705	0.9710	2598	0.9686

Table 2: NDCG at 10 on the Ranking task for all QUBO feature selection methods. The first column refers to all dataset features, the other columns refer to the various solvers used to solve the QUBO feature selection problem. The best results for each QUBO method and dataset are highlighted in bold.

Method	Dataset	All Features		Quantum Solver				Traditional Solver					
				QPU		Hybrid		SA		SD		TS	
		F	NDCG	N	NDCG	N	NDCG	N	NDCG	N	NDCG	N	NDCG
QUBO Correlation	OHSUMED	45	0.3882	35	0.3706	44	0.3659	44	**0.3903**	44	0.3872	26	0.3332
	MQ2007	46	0.4721	43	0.4731	39	0.4720	39	0.4738	28	0.4733	24	**0.4757**
	MQ2008	46	**0.4891**	4	0.4831	21	0.4677	21	0.4657	19	0.4854	25	0.4609
QUBO Boosting	OHSUMED	45	0.3882	9	0.3457	32	0.3632	19	0.3382	40	**0.4002**	23	0.3884
	MQ2007	46	0.4721	43	**0.4760**	37	0.4638	42	0.4632	36	0.4640	35	0.4662
	MQ2008	46	**0.4891**	8	0.4599	8	0.4736	20	0.4852	39	0.4759	34	0.4853
MIQUBO	OHSUMED	45	0.3882	11	0.3685	17	0.3750	17	**0.3942**	6	0.3755	4	0.3882
	MQ2007	46	0.4721	25	**0.4798**	34	0.4722	34	0.4685	34	0.4722	2	0.4721
	MQ2008	46	**0.4891**	1	0.4743	18	0.4791	18	0.4791	18	0.4791	18	**0.4891**

with a lower number of selected features. Comparing the number of selected features with the desired number, i.e., k, reveals that in half of all cases the number of actually selected features is within 10% of k, while the remaining half is split equally in cases where the number of selected features is lower and higher. Since the desired number of features is a penalty, as described in subsection 3.1.4, increasing its strength in the final QUBO problem will allow to better control the resulting number of selected features.

Table 3: Effectiveness of the baseline linear feature selection methods on the Classification and Ranking tasks, measured respectively with classification accuracy and NDCG at 10. Results that are better than or equal to the ones using all the features or QUBO methods are highlighted in bold.

Dataset	ANOVA	Chi2 Test	MI	Pearson Corr.	Linear Boost.	Variance Thr.
waveform-5000	0.8593	0.6893	0.8573	0.8473	0.8533	0.8567
SPECTF	0.8857	0.8571	0.8190	0.8762	0.8762	0.8571
covertype	0.9644	0.9642	0.9612	0.9610	**0.9678**	0.9655
spambase	0.9616	0.9638	0.9631	0.9529	**0.9681**	0.9616
nomao	0.9654	0.9639	0.9656	0.9646	0.9652	0.9648
tecator	0.9306	0.8472	0.9028	0.9028	0.9306	0.9167
USPS	0.9616	0.9616	0.9624	0.9627	0.9631	**0.9652**
isolet	0.9372	0.9406	0.9406	0.9432	0.9389	**0.9444**
Bioresponse	0.7966	0.7948	0.7824	0.7851	0.7922	0.7975
SVHN_small	0.5747	0.5770	0.5606	0.5596	0.5636	0.5616
gisette	0.9748	0.9181	0.9748	0.9700	0.9748	0.9710
OHSUMED	0.3639	0.3475	0.3582	0.3639	0.3900	0.3470
MQ2007	0.4714	0.4614	0.4722	0.4696	0.4703	0.4693
MQ2008	0.4853	0.4788	0.4806	0.4853	0.4831	0.4748

Lastly, Table 3 compares the effectiveness of simple linear feature selection baselines. Such baselines are generally unable to provide better solutions when compared with the QUBO models and the baseline using all features. For the classification task SVC Boosting and Variance Threshold are able to provide better solutions in two cases each, but for different datasets, while for the ranking task none of the baselines exhibits better solution effectiveness.

5.2 Efficiency

This section discusses the computational cost of each step required to use the QPU, following the same structure as Section 3.3.1. The experiments for all Quantum and Classical solvers have been conducted on the same machine to ensure the computational time is comparable and measure the time-to-solution as observed by the local client. For the Traditional Solvers, the time-to-solution of the QUBO problem corresponds to the actual time spent by the solver in finding a solution. For the Quantum Solvers, instead, the time-to-solution has three components: Embedding + Latency + Sampling, as detailed in Table 5. Due to the characteristics of the embedding process, as will be described in this section, for the QPU Solver we consider the time-to-solution as only Latency + Sampling, keeping the Embedding time separate.

Problem Formulation. In order to allow a fair comparison of the time-to-solution of the QUBO solvers one first has to account for the time required to compute the heuristics used by the QUBO problem itself. This is a computationally expensive step that depends both on the number of features and the number of samples in the dataset, roughly requiring to compute $|F|^2/2$ coefficients. This process requires a few seconds until the number of features exceeds 600. For example, computing the QUBO coefficients requires a maximum of ten seconds for `USPS`, while it requires between 4 minutes and one hour for `gisette`, depending on the heuristic. The time required to compute the QUBO coefficients must be taken into account to ensure the time required to solve the problem by the various solvers is put in the right perspective. It should be noted however that the QUBO coefficients must be computed only once and then it can be reused, e.g., deriving the QUBO for each k value, to search for different hyperparameters for the QUBO problem or to for the solver etc. Overall, even though in some cases the time required to generate the QUBO coefficients is quite long, it is always lower than the time required to solve it.

Embedding on the QPU. Table 5 shows the detail of the computational time for the QPU. First, the time required for the minor embedding process varies greatly and while it is in the range of seconds for smaller problems, it can be up to two minutes for problems of a size close to the maximum that can fit on the QPU. Although fully-connected problems are the most difficult and slowest to embed, the embedding only depends on the problem *structure*, not on the specific coefficients. This means that the embedding for a fully-connected QUBO problem of a certain number of variables can be computed once and then it can be reused for any other fully-connected QUBO problem.

Sampling Solutions. Table 4, instead, compares the computational time required by all solvers for the MIQUBO and QUBO-Correlation problems. QUBO-Boosting is omitted for space reasons as it behaves very similarly to QUBO-Correlation. The full results are available in the online material. For both QUBO-Correlation and QUBO-Boosting the QPU has a solution time always lower than both SA and TS, with only SD being faster. In particular, for problems close to the maximum size that can fit on the QPU, SA has a solution time of three times that of the QPU while TS six times. For larger problems, the time gap between SD and TS reduces drastically and Hybrid becomes the fastest solver. It is interesting to notice how for the largest problem of 5000 features SA requires ten times longer than Hybrid. MIQUBO instead shows a different behaviour, with SA comparably much faster and able to show better performance than the QPU for smaller problems. As the problem size grows however the solvers show a similar behaviour as that of QUBO-Correlation, with Hybrid becoming the fastest solver, SA and TS close at twice its solution time and SA being the slowest.

As a general observation it can be seen how the computational time grows as the number of features increases, as can be expected due to the increased problem complexity and values of k explored. The main reason for the increase is however different for classical solvers and the QPU. For classical solvers, the increased solution time is due essentially the greater problem complexity. The QPU instead has a fixed annealing time such that the solution is always returned in constant time for any problem that can fit on its hardware. In these experiments the increased computational time is essentially due to the data transfer, since the coefficients are transmitted to the QPU over the global network and bigger problems will require to transfer more data. Similarly, the Hybrid solver too has a time limit which is a function of the problem size. Note that the constant solution time by the QPU does not offer guarantees regarding its solution quality. For more complex problems it may be needed to sample an increasing number of solutions or to increase the duration of the annealing process. In general, there is no agreed rule on how to choose those settings that impact the underlying physical process. Some bounds have been proven rigorously but only for ideal devices and are, in practice, not applicable to the available QPUs [31].

Table 4: Total time (in seconds) required to solve the k QUBO models with different QUBO solvers, for both the Classification and the Ranking tasks. Due to space limitations, only two of the three methods are reported. However, QUBO-Boosting behaves similarly to QUBO-Correlation, and the complete results can be found in the online material. Results are missing for the QPU when the problem required more qubits than the available ones, in such instances the only Quantum Solver that could be used is Hybrid.

Dataset	F	QUBO-Correlation					MIQUBO				
		Quantum Solver		Traditional Solver			Quantum Solver		Traditional Solver		
		QPU	Hybrid	SA	SD	TS	QPU	Hybrid	SA	SD	TS
waveform-5000	40	39.0	394.4	57.9	7.8	474.6	39.2	385.9	7.5	1.0	78.6
SPECTF	44	48.4	426.3	72.3	9.3	522.4	45.1	421.7	9.5	1.2	86.7
covertype	54	130.1	491.5	17.0	1.9	99.1	127.3	809.9	17.0	1.9	99.0
spambase	57	65.9	487.4	107.1	15.7	598.9	51.5	486.4	17.6	2.0	99.1
nomao	118	100.1	507.5	323.6	54.3	620.9	98.1	512.5	46.6	7.9	101.9
tecator	124	95.6	504.3	374.2	57.7	623.5	90.2	515.1	62.3	8.8	102.3
USPS	256	-	541.1	1174.0	220.4	785.2	-	542.6	138.0	34.6	124.3
isolet	617	-	640.2	6280.7	1260.4	1735.7	-	642.4	838.1	201.6	263.8
Bioresponse	1776	-	1435.1	53978.2	10814.2	10531.4	-	1423.4	11031.8	1727.8	1632.8
SVHN_small	3072	-	3575.4	124041.7	33802.1	32440.0	-	3591.1	23332.5	5455.9	5146.8
gisette	5000	-	8666.3	80062.4	18811.6	18238.1	-	8606.4	78635.7	14210.1	14610.5
OHSUMED	45	154.1	436.8	13.5	1.7	89.4	497.9	435.1	13.9	1.5	89.2
MQ2007	46	472.5	444.5	15.6	1.9	91.5	74.0	445.0	12.8	1.5	91.2
MQ2008	46	122.7	436.7	15.0	1.6	91.7	96.4	443.1	13.0	1.5	91.3

Table 5: Drill down of the time (in seconds) required to solve the QUBO models generated with MIQUBO. The other heuristics behave similarly. The Embedding column refers to the time required to embed the problem on the QPU. The columns under QPU show the time-to-solution as observed by the local client (Total), which corresponds to the QPU time reported in Table 4, splitted between the actual physical annealing process (Sampling) and the latency due to the data transfer as well as further waiting time after the task is queued (Latency). Note that the Latency time is more than one order of magnitude higher than the Sampling time.

Dataset	F	Embedding	QPU		
			Total	Sampling	Latency
waveform-5000	40	7.0	39.2	0.9	38.3
SPECTF	44	5.3	45.1	1.1	44.0
covertype	54	7.9	127.3	1.2	126.1
spambase	57	14.2	51.5	1.0	50.5
nomao	118	209.9	98.1	1.6	96.5
tecator	124	164.6	90.2	1.5	88.7
OHSUMED	45	7.2	497.9	1.0	496.9
MQ2007	46	17.0	74.0	1.0	72.9
MQ2008	46	10.3	96.4	1.2	95.3

Another important aspect apparent from Table 5 is the large difference between the actual duration of the physical annealing process and the time spent by the client waiting for the response. This constitutes another limitation of the current way a QPU can be used as it creates a very large overhead due to the high-latency transfer of the problem data and solution through the global internet network and includes possible waiting time until the QPU becomes available.

Overall, most of the time currently required to solve a fully-connected QUBO problem with a QPU can be eliminated by offering pre-built embeddings and low-latency access.

6 DISCUSSION AND FUTURE WORKS

This work shows the application of currently available Quantum Annealer technology to the feature selection task in classification and ranking problems. The results show that:

(1) Quantum Annealing can be used to solve the feature selection task and the quality of the solutions obtained for both classification and ranking problems, either in terms of accuracy or NDCG, is comparable with the quality obtained with traditional solvers;
(2) for problems able to fit within the number of available qubits, Quantum Annealing requires less time than any other traditional solver, while for very large problems the Hybrid quantum-classical solver is faster than traditional solvers.

From a broader perspective, this study provides evidence that Quantum Annealer technology has evolved to the point that it can be used to tackle real problems. This, in combination with the simple computing model and the easy access to Quantum Annealers in the cloud opens new research directions in the application of this technology to tackle computationally intensive tasks in many IR domains. There is a significant need to identify which are the tasks that fit well in a QUBO formulation or can be efficiently approximated into one by leveraging or developing new heuristics. Many open questions also lie in understanding how to efficiently perform the minor embedding phase, especially for problems that do not have the regular structure of the fully-connected feature selection one. As the technology and tools improve, it is easy to imagine a library of precomputed embeddings being available for problems of particular structures in a similar way as how pretrained machine learning models are. This would completely remove the computational cost of generating the embedding for those cases. Another important direction of improvement is to reduce the effects of network latency, which will be minimized when quantum technology is integrated into the low-latency networks of data centers. Finally, the impact of advanced Quantum Annealer controls and advanced annealing schedule (e.g., reverse annealing [22]), that

really bring the researcher or practitioner close to the physics of the underlying system, can have a strong impact on the solution quality or on the likelihood of finding a good solution but are not yet well understood.

Overall, this work has shown that Quadratic Unconstrained Binary Optimization (QUBO) and Quantum Annealing (QA) are viable options for improving feature selections for both classification and ranking and the above discussion on future perspectives gives an idea of how much room for improvement is already possible to imagine. Therefore, it would be definitely worth if we, as a community, undertake a systematic exploration of these promising research directions, not forgetting that while feature selection is a specific task, for other relevant tasks as well it may be possible to develop a formulation suitable for applying quantum computing approaches. Ranking and classification are central not only to IR but also to several neighbourhood areas, such as natural language processing and recommender systems. Therefore, we could promote some joint effort across these communities, in order to maximize the impact and benefit from cross-fertilization. In this respect, IR has an extremely long tradition in community-wide cooperation on shared research activities, very successfully embodied by large scale evaluation campaigns, as TREC, CLEF, NTCIR and FIRE. It would be extremely valuable if such initiatives take a lead and promote the organization of shared activities for exploring the application of quantum computing to IR, NLP, and RecSys in a comparable and shared way.

7 ACKNOWLEDGMENTS

We acknowledge the CINECA award under the ISCRA initiative, for the availability of quantum computing resources and support. The work was partially supported by University of Padova Strategic Research Infrastructure Grant 2017: "CAPRI: Calcolo ad Alte Prestazioni per la Ricerca e l'Innovazione".

REFERENCES

[1] Steven H. Adachi and Maxwell P. Henderson. 2015. Application of Quantum Annealing to Training of Deep Neural Networks. *CoRR* abs/1510.06356 (2015). arXiv:1510.06356 http://arxiv.org/abs/1510.06356

[2] Tameem Albash and Daniel A Lidar. 2018. Adiabatic quantum computation. *Reviews of Modern Physics* 90, 1 (2018), 015002.

[3] Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. 2018. Quantum boltzmann machine. *Physical Review X* 8, 2 (2018), 021050.

[4] B Apolloni, N Cesa-Bianchi, and D De Falco. 1988. *A numerical implementation of "quantum annealing"*. Technical Report BIBOS-324. Bielefeld TU. Bielefeld-Bochum-Stochastik, Bielefeld. https://cds.cern.ch/record/192546

[5] Christian Bauckhage, Nico Piatkowski, Rafet Sifa, Dirk Hecker, and Stefan Wrobel. 2019. A QUBO Formulation of the k-Medoids Problem. In *Proceedings of "Lernen, Wissen, Daten, Analysen" (CEUR Workshop Proceedings, Vol. 2454)*. 54–63. http://ceur-ws.org/Vol-2454/paper_39.pdf

[6] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, Madeleine Clare Elish, William Isaac, and Richard S. Zemel (Eds.). ACM, 610–623. https://doi.org/10.1145/3442188.3445922

[7] Trevor J. Bihl, Kenneth W. Bauer Jr., and Michael A. Temple. 2016. Feature Selection for RF Fingerprinting With Multiple Discriminant Analysis and Using ZigBee Device Emissions. *IEEE Trans. Inf. Forensics Secur.* 11, 8 (2016), 1862–1874. https://doi.org/10.1109/TIFS.2016.2561902

[8] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. 2020. Next-generation topology of d-wave quantum processors. *arXiv preprint arXiv:2003.00133* (2020).

[9] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[10] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, Bernhard Schölkopf, John C. Platt, and Thomas Hofmann (Eds.). MIT Press, 193–200. https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html

[11] Costantino Carugno, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2022. Evaluating the job shop scheduling problem on a D-wave quantum annealer. *Nature Scientific Reports* 12, 1 (21 Apr 2022), 6539. https://doi.org/10.1038/s41598-022-10169-0

[12] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Comput. Electr. Eng.* 40, 1 (2014), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

[13] Vicky Choi. 2008. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Inf. Process.* 7, 5 (2008), 193–209. https://doi.org/10.1007/s11128-008-0082-9

[14] Maurizio Ferrari Dacrema, Nicolò Felicioni, and Paolo Cremonesi. 2021. Optimizing the Selection of Recommendation Carousels with Quantum Computing. In *RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021*, Humberto Jesús Corona Pampín, Martha A. Larson, Martijn C. Willemsen, Joseph A. Konstan, Julian J. McAuley, Jean Garcia-Gathright, Bouke Huurnink, and Even Oldridge (Eds.). ACM, 691–696. https://doi.org/10.1145/3460231.3478853

[15] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. 2016. What is the Computational Value of Finite-Range Tunneling? *Phys. Rev. X* 6 (Aug 2016), 031015. Issue 3. https://doi.org/10.1103/PhysRevX.6.031015

[16] R Dhanya, Irene Rose Paul, Sai Sindhu Akula, Madhumathi Sivakumar, and Jyothisha J Nair. 2020. F-test feature selection in Stacking ensemble model for breast cancer prediction. *Procedia Computer Science* 171 (2020), 1561–1570. https://doi.org/10.1016/j.procs.2020.04.167 Third International Conference on Computing and Network Communications (CoCoNet'19).

[17] Maurizio Ferrari Dacrema, Tang-Tang Zhou, Riccardo Nembrini, and Paolo Cremonesi. 2021. Quantum Annealing Linear Regression For Collaborative Filtering Recommendations. *2nd European Quantum Technologies Virtual Conference (EQTC), 29 November, 2021* (2021).

[18] Artur J. Ferreira and Mário A. T. Figueiredo. 2012. Efficient feature selection filters for high-dimensional data. *Pattern Recognit. Lett.* 33, 13 (2012), 1794–1804. https://doi.org/10.1016/j.patrec.2012.05.019

[19] Jerome H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232. http://www.jstor.org/stable/2699986

[20] Andrea Gigli, Claudio Lucchese, Franco Maria Nardini, and Raffaele Perego. 2016. Fast Feature Selection for Learning to Rank. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016*, Ben Carterette, Hui Fang, Mounia Lalmas, and Jian-Yun Nie (Eds.). ACM, 167–170. https://doi.org/10.1145/2970398.2970433

[21] Fred W. Glover, Gary A. Kochenberger, and Yu Du. 2019. Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4OR* 17, 4 (2019), 335–371. https://doi.org/10.1007/s10288-019-00424-y

[22] John K. Golden and Daniel O'Malley. 2020. Reverse Annealing for Nonnegative/Binary Matrix Factorization. *CoRR* abs/2007.05565 (2020). arXiv:2007.05565 https://arxiv.org/abs/2007.05565

[23] Nathalie Japkowicz and Mohak Shah (Eds.). 2011. *Evaluating Learning Algorithms: A Classification Perspective.* Cambridge University Press.

[24] Alan Jovic, Karla Brkic, and Nikola Bogunovic. 2015. A review of feature selection methods with applications. In *38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015, Opatija, Croatia, May 25-29, 2015*, Petar Biljanovic, Zeljko Butkovic, Karolj Skala, Branko Mikac, Marina Cicin-Sain, Vlado Sruk, Slobodan Ribaric, Stjepan Gros, Boris Vrdoljak, Mladen Mauher, and Andrej Sokolic (Eds.). IEEE, 1200–1205. https://doi.org/10.1109/MIPRO.2015.7160458

[25] Alexandr Katrutsa and Vadim V. Strijov. 2017. Comprehensive study of feature selection methods to solve multicollinearity problem according to evaluation criteria. *Expert Syst. Appl.* 76 (2017), 1–11. https://doi.org/10.1016/j.eswa.2017.01.048

[26] Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi. 1983. Optimization by Simmulated Annealing. *Sci.* 220, 4598 (1983), 671–680.

[27] Hanjiang Lai, Yan Pan, Yong Tang, and Rong Yu. 2013. FSMRank: Feature Selection Algorithm for Learning to Rank. *IEEE Trans. Neural Networks Learn. Syst.* 24, 6 (2013), 940–952. https://doi.org/10.1109/TNNLS.2013.2247628

[28] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5. https://doi.org/10.3389/fphy.2014.00005

[29] Claudio Lucchese and Franco Maria Nardini. 2017. Efficiency/Effectiveness Tradeoffs in Learning to Rank. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2017, Amsterdam, The Netherlands, October 1-4, 2017*, Jaap Kamps, Evangelos Kanoulas, Maarten de Rijke, Hui Fang, and Emine Yilmaz (Eds.). ACM, 329–330. https://doi.org/10.1145/3121050.3121109

[30] Stefano Marchesin, Alberto Purpura, and Gianmaria Silvello. 2020. Focal elements of neural information retrieval models. An outlook through a reproducibility study. *Inf. Process. Manag.* 57, 6 (2020), 102109. https://doi.org/10.1016/j.ipm.2019.102109

[31] Catherine C. McGeoch. 2020. Theory versus practice in annealing-based quantum computing. *Theor. Comput. Sci.* 816 (2020), 169–183. https://doi.org/10.1016/j.tcs.2020.01.024

[32] Massimo Melucci. 2015. *Introduction to Information Retrieval and Quantum Mechanics.* The Information Retrieval Series, Vol. 35. Springer. https://doi.org/10.1007/978-3-662-48313-8

[33] Riccardo Nembrini, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2021. Feature Selection for Recommender Systems with Quantum Computing. *Entropy* 23, 8 (2021), 970. https://doi.org/10.3390/e23080970

[34] Hartmut Neven, Vasil S. Denchev, Geordie Rose, and William G. Macready. 2012. QBoost: Large Scale Classifier Training with Adiabatic Quantum Optimization. In *Proceedings of the 4th Asian Conference on Machine Learning, ACML 2012, Singapore, Singapore, November 4-6, 2012 (JMLR Proceedings, Vol. 25)*, Steven C. H. Hoi and Wray L. Buntine (Eds.). JMLR.org, 333–348. http://proceedings.mlr.press/v25/neven12.html

[35] Gintaras Palubeckis. 2004. Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem. *Ann. Oper. Res.* 131, 1-4 (2004), 259–282. https://doi.org/10.1023/B:ANOR.0000039522.58036.68

[36] Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. 2005. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 8 (2005), 1226–1238. https://doi.org/10.1109/TPAMI.2005.159

[37] Alberto Purpura, Karolina Buchner, Gianmaria Silvello, and Gian Antonio Susto. 2021. Neural Feature Selection for Learning to Rank. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12657)*, Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani (Eds.). Springer, 342–349. https://doi.org/10.1007/978-3-030-72240-1_34

[38] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.* 13, 4 (2010), 346–374. https://doi.org/10.1007/s10791-009-9123-y

[39] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[40] Irene Rodríguez-Luján, Ramón Huerta, Charles Elkan, and Carlos Santa Cruz. 2010. Quadratic Programming Feature Selection. *J. Mach. Learn. Res.* 11 (2010), 1491–1516. http://portal.acm.org/citation.cfm?id=1859900

[41] Noelia Sánchez-Maroño, María Caamaño-Fernández, Enrique F. Castillo, and Amparo Alonso-Betanzos. 2006. Functional Networks and Analysis of Variance for Feature Selection. In *Intelligent Data Engineering and Automated Learning - IDEAL 2006, 7th International Conference, Burgos, Spain, September 20-23, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4224)*, Emilio Corchado, Hujun Yin, Vicente J. Botti, and Colin Fyfe (Eds.). Springer, 1031–1038. https://doi.org/10.1007/11875581_123

[42] Jean Tague-Sutcliffe and James Blustein. 1994. A Statistical Analysis of the TREC-3 Data. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994 (NIST Special Publication, Vol. 500-225)*, Donna K. Harman (Ed.). National Institute of Standards and Technology (NIST), 385.

[43] Kari Torkkola. 2003. Feature Extraction by Non-Parametric Mutual Information Maximization. *J. Mach. Learn. Res.* 3 (2003), 1415–1438. http://jmlr.org/papers/v3/torkkola03a.html

[44] Sagar Uprety, Dimitris Gkoumas, and Dawei Song. 2020. A Survey of Quantum Theory Inspired Approaches to Information Retrieval. *ACM Comput. Surv.* 53, 5 (2020), 98:1–98:39. https://doi.org/10.1145/3402179

[45] Hayato Ushijima-Mwesigwa, Christian F. A. Negre, and Susan M. Mniszewski. 2017. Graph Partitioning using Quantum Annealing on the D-Wave System. *In Proceedings of the Second International Workshop on Post Moores Era Supercomputing (PMES '17)* abs/1705.03082 (2017). arXiv:1705.03082 http://arxiv.org/abs/1705.03082

[46] Cornelis Joost van Rijsbergen. 2004. *The geometry of information retrieval.* Cambridge University Press.

[47] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. 2013. OpenML: networked science in machine learning. *SIGKDD Explor.* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[48] Dennis Willsch, Madita Willsch, Hans De Raedt, and Kristel Michielsen. 2020. Support vector machines on the D-Wave quantum annealer. *Comput. Phys. Commun.* 248 (2020), 107006. https://doi.org/10.1016/j.cpc.2019.107006

[49] Zilin Zeng, Hongjun Zhang, Rui Zhang, and Chengxiang Yin. 2015. A novel feature selection method considering feature interaction. *Pattern Recognit.* 48, 8 (2015), 2656–2666. https://doi.org/10.1016/j.patcog.2015.02.025