Discovering Characterization Rules from Rankings

Ansaf Salleb-Aouissi
CCLS
Columbia University
New York, NY 10115
ansaf@ccls.columbia.edu

Bert Huang
Computer Science Department
Columbia University
New York, NY 10027
bert@cs.columbia.edu

David Waltz
CCLS
Columbia University
New York, NY 10115
waltz@ccls.columbia.edu

Abstract

For many ranking applications we would like to understand not only which items are top-ranked, but also why they are top-ranked. However, many of the best ranking algorithms (e.g., SVMs) are black boxes that give little information about the factors for their rankings. We describe and demonstrate a new approach that can work in conjunction with any ranking algorithm to discover explanations for the items at the top of the rankings. These explanations are in the form of rules expressed as boolean combinations of attribute-value expressions. These rules are discovered by contrasting attributes of items drawn from both the top and bottom of a ranking list, looking for items that have high leverage, corresponding to rules with broad coverage and sharp differentiations. We include empirical results to demonstrate the utility of our method.

1. Introduction

Ranking problems arise in a wide range of real world applications where an ordering on a set of examples is preferred to a classification model. These applications include collaborative filtering, information retrieval and recommender systems. In most of these applications, the user typically focuses and acts only on the top part of the ranking list. For example, when a browser returns thousands of relevant pages to a query, usually only the first few results are exploited by the user. In some other applications, for instance, ranking components of a system by susceptibility to failure, the user is generally interested in acting only on the top n-percent of components and has no interest in the ordering of the other components.

One of the main limitations of ranking models is the lack of intelligibility of the results. Typically no insights on the factors of a ranking or explanations are given to the user – the results produced must be taken on faith. Our work was initially motivated by a real application related to ranking power grid electrical components according to their likelihood to failures. From our practical experience, decision-makers are understandably more confident acting on high-ranked items if they can be given factors for the rankings than if they are simply given an ordered list of examples.

We address here the problem of interpretability of a ranking list of examples by learning symbolic interpretations with emphasis on the top portion of the ranking. Our goal is not at all to uncover the ranking function used by a black-box ranking method but to analyze the top portion of the ranking list. To the best of our knowledge no work has been done yet on getting explanations from rankings. The general schema of our methodology is as follows:

- Rank examples using any supervised ranking system.
- Create two subsets of the ranking data, consisting of the top n and bottom m items of the ranking. Typically $n = m$, and $||n + m|| = 5 - 20\%$ of the total data.
- Extract the important characteristic properties by analyzing attribute patterns in the top and bottom subsets.

We have developed *YRank*, an algorithm designed to learn interpretable characterization rules of the top of a ranking list. Note that the bottom portion of the ranking list is used to select the most powerful rules that describe top examples, and sharpen these rules. *Interestingness* is assessed in our framework with the *leverage* measure that combines high coverage and characterization power. We have applied our algorithm to several datasets. Our empirical results show the utility of our methodology with regard to the interpretability of a ranking list of examples. The contributions of the paper are as follow:

1. We argue and show that learning characterization rules is an effective approach to understanding the combinations of attributes and values that characterize the top portion of a ranking list and differentiate it from the bottom of the ranking list.

2. We present an effective method for computing characterization rules that use only the top and bottom of

rankings and demonstrate its effectiveness on several datasets.

The paper is organized as follows: In Section 2, we give the related work of ranking models and interpretable models. Our framework is described in Section 3. Section 4 is devoted to the experiments. We conclude in Section 5.

2. Related Work

2.1 Ranking models

There has been an increasing interest in learning ranking models [1]. Freund et al. [8] proposed *RankBoost*, a boosting algorithm that combines many "weak" rankings into a single accurate ranking. *SVM-Rank* was proposed by Joachims [14] and applied to improve information retrieval problems by using click-through data as feedback from users in search engines. Long and Servedio developed a boosting method [16] named *Martingale boosting* for ranking. In addition to *pure* ranking methods that learn *ranking functions*, some *classification functions* that output scores have also shown excellent performances in ranking [24]. For instance, it turned out that SVM classifiers are good rankers as well [12]. Typically, the SVM produces a classifier that labels examples, then thresholds the outputs. Instead, one can rank the examples by how strongly the classifier predicts the class of each example.

2.2 Interpretable models

Interpretability of Machine Learning and Data Mining models has been recognized as one of the main challenges in the field [13]. In many practical applications, classification accuracy is no longer the only goal; interpretability is critical as well. Quite a lot of recent work has addressed the problem of extracting explanations from classifiers. For instance, there are many approaches for extracting explanations from SVM (e.g., [4, 9]). Most of these works use the SVM's "support vectors" to produce rules.

The machine learning and data mining literature is rich with numerous approaches for learning interpretable models spanning descriptive generalization [19], decision trees [21], association rules [2, 22], subgroup discovery [15], and characteristic patterns [23]. These methods share the goal of extracting intelligible patterns, rules and models from the data set. One can distinguish our goal of *explanatory learning* from *discriminative learning*, such as in decision trees [11]. For decision trees, discriminative rules are formed via paths from the root to the leaves, which contain the class. That is, they discriminate between the classes present in the learning set. Although such rules are interpretable, a large amount of data is needed to get a stable decision tree, whereas we argue that a better characterization can be obtained using only the top of the ranking list.

2.3 Why explanatory rules provide interpretability

The explanatory descriptions we are using are in general *maximal* conjunctions of attribute-relation-value expressions, whereas discrimination rules are usually *minimal* descriptions [19]. For example (taken from [7]), suppose that we want to learn the decision mechanism of a "banks coin-sorting machine" for a given number of different coins (positive examples). It is most important to train the system to recognize every single coin the machine is supposed to accept (i.e., to characterize), and not so critical to discriminate these coins from all of the infinite different kinds of coins that should be rejected. However, if available, one can use other coins (e.g., some faked and foreign coins) as negative examples, to keep only the most characteristic descriptions, i.e., to eliminate items that are common to both the bottom and top of a ranking, and thus true but uninformative. We believe and show in Section 4.2 that long descriptions constituted by strong properties discovered by a characterization approach are more likely to be useful and actionable. That is, if we would like to act on the top ranked elements, we would like to know as many details as possible about those elements.

3. From a ranking list to interpretability

Our approach is a two-step process. The first step consists of ranking the examples using any Machine Learning ranker. The second step concerns the extraction of explanations from a ranked list of objects.

3.1 Ranking framework

We address the problem of *supervised ranking* of data. The term *ranking* refers to the process of taking a collection of data and ordering it in a meaningful and useful order. Supervised ranking outputs such an ordered set using the features and guided by the label assigned to each object.

More formally, we would like to order a set of examples $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_1, \ldots, x_n$ are vectors of features describing a set of examples, and each example is given a label $y_i \in \{+1, -1\}$. We will denote by $\mathcal{X}^+$ and $\mathcal{X}^-$ the set of positive and negative examples respectively. Ideally, we want to learn a scoring hypothesis h that would allow us to rank all the positive examples above all the negative ones. That is, $\forall x_i \in \mathcal{X}^+, \forall x_j \in \mathcal{X}^-, h(x_i) > h(x_j)$.

We use receiver order characteristic (ROC) curves [5] to analyze the ranking results, since they provide a good way of measuring the quality of a ranking. when the only ground truth we have is whether or not each data point belongs on the top of the ranking (labeled $+1$) or on the bottom (labeled -1). ROC is essentially normalized by the class cardinality.

	Rank	Serial#	Age	Size	Manufacturer
Top	1	15B25	2	500	B
	2	13B28	8	500	B
	3	58C25	12	1000	C
	4	88A25	1	500	A
	5	18B22	17	500	B
			⋮		
Bottom	96	63A11	27	500	A
	97	12A25	2	2000	A
	98	15A54	8	2000	A
	99	55A95	12	2000	A
	100	41B77	25	2500	B

Table 1. A toy example of a ranking of components of an electrical system.

Example. Consider a list of 100 electric components (Table 1). Each component is described by its serial number, age, size and manufacturer, and is labeled according to its failure status (label=1 for failed, -1 otherwise). A ranking allows us to order the components according to susceptibility of failure. The *top* of the ranking would have the components that are the most susceptible to failure while components at the *bottom* of the ranking are less prone to failure. Thus, the domain expert can focus on the n top components and act on them, e.g., by scheduling inspections/replacements [1].

3.2 Interpretable explanations

Given a "good ranking" list produced by the previous step, getting interpretable explanations consists of characterizing the top (highly ranked) examples in the ranking list. The bottom examples of the ranking list are used for contrast, to make sure that we identify and retain the most characteristic rules of the top examples. By doing so, we group the examples into three sets, where the most "pure" examples, i.e., the "very positive" and "very negative" examples are on the top and bottom of the ranked list respectively. We focus on the top and the bottom of a ranking rather than simply contrasting the positive to the negative class because we are only interested in the top part of the ranking list; this is generally the actionable information for the user. Also, in practice, some examples are considered negative examples while they are actually unknown. This occurs in many real-life applications. In the previous example, we are not sure whether the negative examples are truly negative; Although these components are considered as negative examples because they have not failed yet, they could fail soon. The ranking function would not rank these examples deep in the top or bottom classes but rather in the middle. The approach we suggest here would discard such uncertain examples and focus only on the most certain and thus valuable ones. Once we focus on the ranking extremities, we look for the set of interesting rules, characterizing the top of the list, of the form:

$$R : \text{Concept} \rightarrow \text{Property},$$

where *Property* is an attribute-value pair and *Concept* is either the concept "in the top" or "in the bottom"[2]. A rule is interesting if it has enough *coverage*; that is the proportion of example having this property in the top ranked part of the ranking high enough w.r.t. a minimum coverage threshold. More formally, the coverage is defined by:

$$\text{Coverage}(R) = \frac{|\{x \in \text{Concept} \wedge \mathcal{V}_p(x) = \text{true}\}|}{|x \in \text{Concept}|}.$$

Where p is the property and the notation $\mathcal{V}_p(x)$ is a Boolean function such that for an example x, we have $\mathcal{V}_p(x) = \text{true}$ or false which means that the property p may be satisfied by x or not.

We also assess the importance of properties by using other statistical measures such as the *leverage* measure [20]. The reason we chose this statistical measure[3] is that it combines high characteristic power with the capture of the most highly associated properties (high coverage). The leverage has been used in other learning tasks and is called also *novelty* (e.g., [23] in learning characteristic rules). The leverage of the rule above is given by (P stands for probability):

$$\begin{aligned}\text{Leverage}(R) &= P(\text{Property} \wedge \text{Concept}) - \\ &\quad P(\text{Property}) \times P(\text{Concept}).\end{aligned}$$

The leverage measure evaluates the proportion of additional examples covered by both the left-hand side and right-hand side of the rule above those expected if both sides of the rule were independent of each other. Obviously, we have: $-0.25 \leq \text{Leverage}(R) \leq +0.25$.

A property is interesting for a given concept if it has a strongly positive or negative leverage. A strongly positive value indicates a strong association between the property and the concept, while a strongly negative value indicates a strong association between the property and the negation of the concept. We estimate the leverage of a rule by:

$$\frac{|\{x \in \text{Concept} \wedge \mathcal{V}_p(x) = \text{true}\}|}{|T \cup B|} - \frac{|\{x \in (T \cup B) \wedge \mathcal{V}_p(x) = \text{true}\}|}{|T \cup B|} \times \frac{|x \in \text{Concept}|}{|T \cup B|},$$

where *Concept* is either T (top set) or B (bottom set).

[1] Our actual rankings are based on 100-300 attributes for each component.

[2] Note that rules involving the concept "in the bottom" are by-products in our approach as we aim at characterizing the top of the ranking..

[3] One can use other evaluation measures such as entropy, purity, or Laplace estimate [10] to assess the interestingness of rules.

	Property	Coverage_top	Leverage_top	Coverage_bottom	Leverage_bottom
Top	Manufacturer=B	0.6	0.1	0.2	-0.1
	Size=[500,1000)	0.8	0.15	0.2	-0.15
	Manufacturer=B AND Size=[500,1000)	0.6	0.15	0	-0.15
	Age=(-inf,3) AND Size=[500,1000)	0.4	0.10	0	-0.10
Bottom	Size=[2000,2500)	0	-0.15	0.6	0.15
	Manufacturer=A	0.2	-0.15	0.8	0.15
	Age=[25,+inf)	0	-0.10	0.4	0.10
	Manufacturer=A AND Size=[2000,2500)	0	-0.15	3	0.15

Table 2. Set of properties extracted for the toy example. Manufacturer A seems to make rather good large-size components, while manufacturer B makes bad small components.

Example. Consider the ranked list of electric components illustrated in Table 1. The main extracted patterns shown in Table 2 help to identify which properties are responsible for failures. It can be extremely important to find patterns in the attributes of highly ranked items, for instance to realize that particular components built during some range of dates by a particular manufacturer are disproportionately responsible for failures. The ultimate goal is to help the domain expert set policies for purchasing the most reliable components, schedule inspections, etc.

Algorithm 1: YRank pseudo-code

```
Input:
- a ranking list of examples $\mathcal{L}$, a coverage threshold MinCov, a
leverage threshold MinLev, Top and Bottom percentages
Output:
- 2 sets of properties $\mathcal{P}_T$ and $\mathcal{P}_B$, a set of histograms $\mathcal{H}$
$T \leftarrow \{\text{examples in top of } \mathcal{L}\}$
$B \leftarrow \{\text{examples in bottom of } \mathcal{L}\}$
$\mathcal{H} \leftarrow \emptyset$ , $\mathcal{P}_T \leftarrow \emptyset, \mathcal{P}_B \leftarrow \emptyset, \mathcal{C}_1 \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset, i = 1$
foreach attribute do
    foreach value do
        $p \leftarrow$ (attribute $=$ value)
        $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup \{p\}$
    $h =$ Histogram(attribute)
    $\mathcal{H} = \mathcal{H} \cup h$
while $\mathcal{C}_i \neq \emptyset$ do
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{p \in \mathcal{C}_i$ / coverage($p$) $\geq$ MinCov$\}$
    $\mathcal{C}_{i+1} =$ generate_properties($\mathcal{C}_i$)
    $i = i + 1$
$\mathcal{P}_T \leftarrow \mathcal{P}_T \cup \{p \in \mathcal{P}/$Leverage(Top $\rightarrow p$) $\geq$ MinLev$\}$
$\mathcal{P}_B \leftarrow \mathcal{P}_B \cup \{p \in \mathcal{P}/$Leverage(Bottom $\rightarrow p \geq$ MinLev$\}$
return $\mathcal{P}_T, \mathcal{P}_B, \mathcal{H}$
```

The pseudo-code of YRank is given in Algorithm 1. The aim of YRank is to uncover the set of the most important rules that lead a supervised ranking algorithm to rank some examples above some others. The algorithm explores the search space of possible properties by contrasting top and bottom parts of the ranking. The terms $\mathcal{P}_T$ and $\mathcal{P}_B$ are used to denote the set of properties (right-hand sides of the rules) for *top* and *bottom* respectively. For a better visualization of the coverage of the properties in *top* and *bottom*, our code outputs also a histogram for each attribute giving the relative frequency of its various values. This allows us to visualize the most contrasting single properties. We use a variant of the classical level-wise framework [18] for learning all interesting properties efficiently by using the anti-monotonicity property of the coverage measure. That is, if a property does not have a sufficient coverage, no conjunction of properties including that property will have enough coverage w.r.t. the minimum coverage threshold, since coverage only decreases when adding conditions. YRank starts with single properties that have sufficient coverage. The function *generate_properties* constructs properties of of size $k+1$ by joining properties of size k that have $k-1$ properties in common [2]. This ensures that we will get conjunctions of properties of size exactly $k+1$. At each round, only the properties having enough coverage are kept in the set of properties $\mathcal{P}$. We try several values of top and bottom percentages in order to select the best sizes for *top* and *bottom*, the sizes that lead to the highest number of interesting properties and the highest leverages.

4. Experiments

We implemented YRank[4] in Python and conducted an experimental evaluation of our algorithm on several benchmarks. We have used first SVMLight[5] to train Support Vector Machines on the datasets in order to get the ranking lists.

4.1 A synthetic dataset

To verify whether YRank is catching the right attributes, we randomly generated a synthetic dataset of 1000 examples each described by 50 features such that $X \in \{-1, 1\}^{50}$. Class labels were assigned as follows:

$$Y = \text{sign}(\textstyle\sum_{k=1}^{k=11} X_k)$$

YRank succeeded in finding those attributes with a minimum Leverage of 0.08 by focusing only on the top 5% and

[4] Code & datasets available at http://www1.ccls.columbia.edu/~ansaf/

[5] http://svmlight.joachims.org/

bottom 5% of the ranking list. It was not possible to uncover this set of properties by using the full dataset (where top (resp. bottom) represents all positive (resp. negative) examples) with the same parameters until we decreased the Leverage to a very low value. This means that we have more characteristic power in the top and bottom of the ranking list than in the full dataset. See discussion in Section 4.3.

4.2 Atherosclerosis dataset

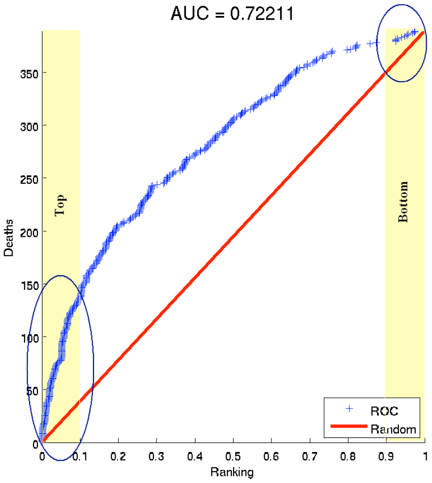

Figure 1. Atherosclerosis ROC Curve. The X-axis represents the ranking in proportions.

In this section, we describe the experiments we conducted on a medical dataset[6] in the context of the Stulong project. The dataset concerns a twenty year study of the risk factors for atherosclerosis in a population of 1,419 middle aged men. We use a compiled dataset [17] with the goal of identifying the main factors for this disease. The attributes used in the dataset are given in Table 5 in the Appendix. The ranking target attribute used is “death”. Figure 1 shows the ROC curve for the learning results with top 10% and bottom 10% emphasized and where *top* gathers the sickest patients and *bottom* the healthiest patients. We have used YRank with different values of *top* and *bottom* and kept the pair (*top*=5% and *bottom*=5%) giving the highest number of good properties. The results are reported in Table 4 and the associated histograms are in the Appendix Figure 3.

Stulong data has already been used in a discovery challenge and has been the subject to many publications [17]. Atherosclerosis factors are known to be mainly tobacco consumption and duration, overweight, and low physical activity while there is no evidence on the impact of alcohol consumption. All these factors have been uncovered by our approach as shown in Table 3, and in the Appendix Figure 3. We show that there are many strong properties characterizing the top 5% of the ranking list discovered by Yrank. For example property 1, RSK_TOBA=1, is a perfect property since it covers all the patients at the top of the ranking list. Other interesting properties include for instance, Toba_duration=20, education =0 and TOBA_CONSO=1.25.

To compare to other symbolic learning methods, we used other classifiers that output interpretable classification rules such as the Ripper classification rule induction algorithm [6] and C4.5 decision trees [21]. The goal was to classify the top and the bottom patients, each of which constituted of 70 patients. The results are very short classification rules based on property 1 that Yrank discovered. No more details are given about the population of patients at the top of the ranking. The output results are as follows:

- Ripper rule learner:

```
  RSK_TOBA >= 1 → Concept=Top  (70/0)
```

where (70/0) expresses (#patients Top/#patients Bottom)

- C4.5 Decision tree:

```
      RSK_TOBA <= 0:  Bottom  (70)
        RSK_TOBA > 0:  Top  (70)
```

where (70) expresses the number of patients in Bottom/Top. Obviously, RISK_TOBA is the most discriminative property that both classifiers chose and there was no need for the classifiers to use any other property to discriminate the top and the Bottom. However, this property is not actionable and thus does not help the domain-expert while other strong properties discovered by our characterization method are quite interpretable and actionable. That is, if we want to act on the top ranked patients, we would like to know as many details as possible about those patients.

4.3 More experiments

	Size_prop = 1		**Size_prop** = 2	
	top+bottom	Full dataset	*top+bottom*	Full dataset
Atheroscl.	20	2	463	63
Australian	15	7	135	71
Heart	21	11	186	78
Synthetic	26	2	975	180

Table 4. Number of rules discovered from full datasets vs. Top 5% and Bottom 5%.

We compared the number of interesting properties w.r.t. the leverage measure when we use the top and the bottom examples, *versus* when we use the full dataset. Table 4 compares the number of interesting properties for these 2 cases on 4 datasets: the atherosclerosis dataset, the synthetic dataset and 2 other datasets from the UCI repository

[6] http://euromise.vse.cz/STULONG

	Number	Property	Freq_top	Cov_top	Lev_top	Freq_bottom	Cov_bottom	Lev_bottom
TOP	1	RSK_TOBA=1	70	1.00	0.25	0	0.00	-0.25
	2	TOBA_DURA=20	67	0.96	0.24	1	0.01	-0.24
	3	EDUCATION=0	60	0.86	0.21	2	0.03	-0.21
	4	ICT =0 AND EDUCATION=0	60	0.86	0.21	2	0.03	-0.21
	5	TOBA_DURA=20 AND EDUCATION=0	57	0.81	0.2	0	0.00	-0.2
	6	TOBA_DURA=20 AND RSK_TOBA=1 AND EDUCATION=0	57	0.81	0.2	0	0.00	-0.2
	7	BIRTH_YEAR=[25,30)	52	0.74	0.18	2	0.03	-0.18
	8	RSK_HYPE=1	50	0.71	0.18	1	0.01	-0.17
	9	TOBA_CONSO=1.25	37	0.53	0.12	2	0.03	-0.12
	10	TOBA_CONSO=0.85	33	0.47	0.1	5	0.07	-0.1
	11	HT=0	70	1.00	0.1	41	0.59	-0.1
	12	ACTIV_JOB=3	29	0.41	0.09	4	0.06	-0.09
	13	RSK_OBES=1	28	0.40	0.09	4	0.06	-0.09
	14	TIME_JOB=6	24	0.34	0.06	7	0.10	-0.06
	15	MARIT_STAT=0	21	0.30	0.06	3	0.04	-0.06
	16	SYST=[160,180)	18	0.26	0.06	0	0.00	-0.06
	17	RSK_FAMI=1	19	0.27	0.06	3	0.04	-0.06
	18	DIAST=[100,120)	14	0.20	0.05	0	0.00	-0.05
BOTTOM	19	ALCO_CONS=[1.10,1.20)	23	0.33	-0.05	37	0.53	0.05
	20	SYST=[120,140)	12	0.17	-0.05	26	0.37	0.05
	21	MARIT_STAT=1	49	0.70	-0.06	67	0.96	0.06
	22	SYST=[100,120)	2	0.03	-0.06	18	0.26	0.06
	23	RSK_FAMI=0	50	0.71	-0.06	67	0.96	0.06
	24	TOBA_CONSO=0.5	0	0.00	-0.07	20	0.29	0.07
	25	TIME_JOB=5	36	0.51	-0.08	58	0.83	0.08
	26	RSK_OBES=0	41	0.59	-0.09	66	0.94	0.09
	27	HT=1	0	0.00	-0.1	29	0.41	0.1
	28	ACTIV_JOB=1	16	0.23	-0.12	50	0.71	0.12
	29	TOBA_CONSO=0	0	0.00	-0.13	36	0.51	0.13
	30	BIRTH_YEAR=[35,40)	4	0.06	-0.17	51	0.73	0.17
	31	RSK_HYPE=0	15	0.21	-0.19	69	0.99	0.19
	32	EDUCATION=1	10	0.14	-0.21	68	0.97	0.21
	33	MARIT_STAT=1 AND RSK_HYPE=0 AND RSK_OBES=0	3	0.04	-0.21	62	0.89	0.21
	34	MARIT_STAT=1 AND RSK_HYPE=0	5	0.07	-0.22	66	0.94	0.22
	35	RSK_TOBA=0	0	0.00	-0.25	70	1.00	0.25

Table 3. List of some of the properties discovered in atherosclerosis dataset as generated by YRank.

[3]. For this experiment, we have used MinLeverage=0.08, *top*=*bottom*=5% and a size of properties $Size\_prop \leq 2$. The results show that we get more interesting properties with a high leverage when we consider the top and bottom of the ranking list than when we use the full dataset.

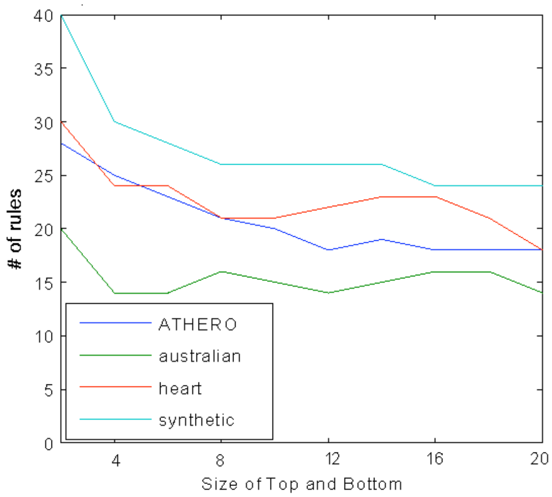

Figure 2. # of rules versus (top+bottom)%.

We conducted other experiments with different top and bottom values. They have shown that as we increase $||T + B||$, we get less interesting properties (Figure 2). We do lose characteristic power when we include data that is less pure than the top and bottom.

We also work with other proprietary data in smart grid applications, for which this framework has been initially derived. The goal is to rank over 30,000 electric components (transformers) according to their susceptibility to failure. Our actual rankings are based on 100-300 attributes for each component. The intelligible rules we have extracted have proven useful for directing the actions of domain experts, and importantly have helped give the domain experts confidence in the correctness of the ranking results.

5. Conclusion and Future Work

This paper describes a simple yet powerful approach to make supervised ranking interpretable, specifically the top portion of a ranking. The underlying idea is to focus on the top and bottom portions of the ranking to uncover the main characteristic properties of the highly ranked examples. This can be useful for the practitioner to direct actions on the top-ranked items and understand the model. Our algorithm needs a ranked list of objects as input, and is independent of the learning methodology was used to rank the objects. In future work, we would like to integrate the ability to interpret properties into the ranking algorithm itself.

Acknowledgments

Thanks to Sergey Sigelman for YRank implementation. This work has been partly supported by a research contract from Consolidated Edison New York.

References

[1] S. Agarwal, C. Cortes, and R. Herbrich, editors. *Proceeding of the NIPS 2005 Workshop on Learning to Rank*, December 2005.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994.

[3] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[4] N. Barakat and A. P. Bradley. Rule extraction from support vector machines: Measuring the explanation capability using the area under the roc curve. In *ICPR (2)*, pages 812–815, 2006.

[5] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[6] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[7] P. Davidsson. Integrating models of discrimination and characterization for learning from examples in open domains. In *IJCAI (2)*, pages 840–845, 1997.

[8] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *ICML '98*, pages 170–178, 1998.

[9] G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *KDD '05*, pages 32–40, New York, NY, USA, 2005. ACM Press.

[10] J. Furnkranz and P. Flach. An analysis of rule evaluation metrics. In *In ICML'03*, pages 202–209. AAAI Press, January 2003.

[11] D. Gamberger and N. Lavrač. Generating actionable knowledge by expert-guided subgroup discovery. In *In PKDD'02*, pages 163–174. Springer-Verlag, 2002.

[12] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression.* MIT Press, Cambridge, MA, 2000.

[13] H. Hirsh. Data mining research: Current status and future opportunities. *Statistical Analysis and Data Mining*, 2008.

[14] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[15] N. Lavrač, B. Cestnik, D. Gamberger, and P. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning*, 57(1-2):115–143, 2004.

[16] P. M. Long and R. A. Servedio. Martingale boosting. In *In COLT 2005*, volume 3559 of *Lecture Notes in Artificial Intelligence*, pages 79–94. Springer, 2005.

[17] N. Lucas, J. Azé, and M. Sebag. Atherosclerosis Risk Identification and Visual Analysis. In *ECML/PKDD 2002 Discovery Challenge Workshop program*, 2002.

[18] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledgediscovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997.

[19] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Springer, Berlin, Heidelberg, 1984.

[20] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, 1991.

[21] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[22] A. Salleb-Aouissi, C. Vrain, and C. Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In *IJCAI*, pages 1035–1040, 2007.

[23] T. Turmeaux, A. Salleb, C. Vrain, and D. Cassard. Learning characteristic rules relying on quantified paths. In L. N. et al., editor, *In PKDD'03*, pages 471–482. Springer–Verlag, Lecture Notes in Computer Science, Sept 2003.

[24] H. Yu. Svm selective sampling for ranking with application to data retrieval. In *In KDD '05*, pages 354–363, New York, NY, USA, 2005. ACM.

Appendix

Attribute	Type	Description
ICO	C	Identification of a patient
ACTIV_JOB	C	Physical activity in a job. 1:sits, 2: stands, 3:walks, 4:carries heavy loads,5: not stated
ACTIV_AFT	C	Physical activity after a job. 1: sits, 2: moderate activity, 3:great activity, 4: not stated
TRANSP_JOB	C	Transport to go to work. 1:on foot, 2: by bike, 3:public means of transport, 4: by car, 9: not stated
TIME_JOB	C	Time to get to work. 5: half hour, 6: 1 hour, 7: 2 hours, 8: ¿2 hours, 9:not stated
BIRTH_YEAR	N	Year of birth
ENTRY_YEAR	N	Year of entry into the study
ALCO_CONS	N	Alcohol consumption
TOBA_CONS	N	Tobacco consumption
TOBA_DURA	N	Smoking duration
MARIT_STAT	C	Marital status. 1:married, 0: not married
EDUCATION	C	Reached education. 1: university, 0: not university
IM	C	Myocardial infarction
ICT	C	Ictus
HT	C	Hypertension
HTL	C	Medicines in HT
DIAB	C	Diabetes
DIABD	C	Diet in DIAB
HYPL	C	Hyperlipidemia
HYPLL	C	Medicines in hyperlipidemia
MOC_SUC	C	Urine sugar
MOC_ALB	C	Urine albumen
BOLHR	C	Chest pain
CHLST	N	Cholesterol in mg%
TRIGL	N	Triglycerides in mg%
SYST	N	Blood pressure systolic
DIAST	N	Blood pressure diastolic
HEIGHT	N	Height (cm)
WEIGHT	N	Weight (kg)
BMI	N	Body Mass Index
TRIC	N	Skin fold triceps (mm)
SUBSC	N	Skin fold subscapularis (mm)
RSK_FAMI	C	Family risk
RSK_OBES	C	Obesity risk
RSK_TOBA	C	Smoking risk
RSK_HYPE	C	Hypertension risk
RSK_CHOL	C	Cholesterol risk
GROUP	C	Normal, Risk, Pathological
DEATH	C	Patient dead or not

Table 5. Attributes of the atherosclerosis table. The type "C" stands for categorical and "N" for numerical.

Histograms in Figure 3 show for each of the attributes TOBA_CONSO, TOBA_DURA, SYST and DIAST the coverage of their various values in Top (blue) and Bottom (red) of the ranking. High tobacco consumption during a long period is more characteristic of the top than the bottom of the ranking list and so are high systolic and diastolic blood pressures.

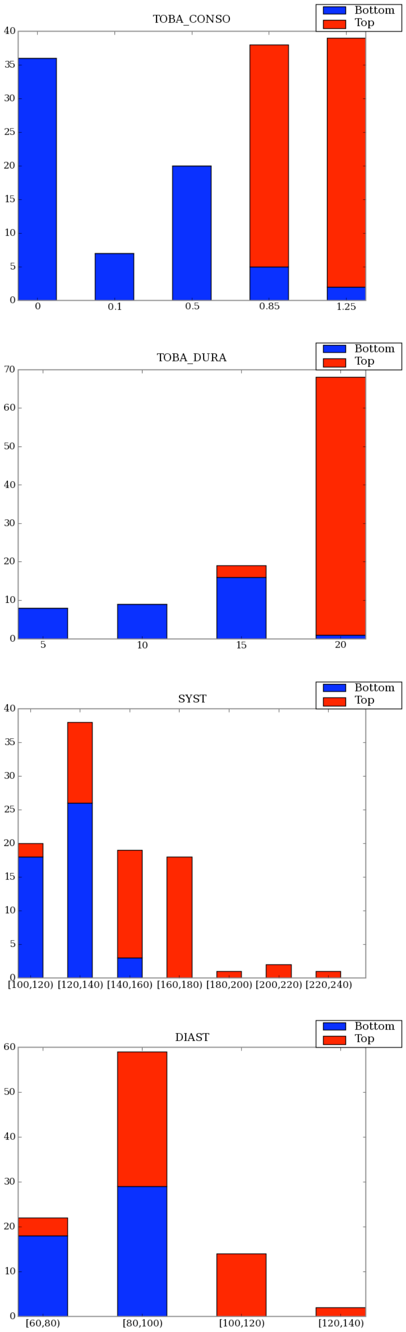

Figure 3. Histograms of some attributes as extracted by YRank.