Sparse Oblique Decision Trees: A Tool to Interpret Natural Language Processing Datasets

Suryabhan Singh Hada and Miguel Á. Carreira-Perpiñán
Dept. of Computer Science and Engineering, University of California, Merced, CA, USA
Email: {shada,mcarreira-perpinan}@ucmerced.edu

***Abstract*—Natural language processing datasets, for example for document classification or sentiment analysis, are characterized by sparse, high-dimensional feature vectors, often based on bag-of-words approaches. Such datasets contain a wealth of information not just about the predictive task in question, but also about the language itself, and it is of interest to do data mining on such data. While one way to do this is to use standard exploratory data analysis techniques such as clustering or dimensionality reduction, here we propose a different approach, which can be used if we have access to a labeled dataset. The idea is to use sparse oblique decision trees, a type of interpretable model having the structure of a decision tree but where the decision nodes use hyperplanes involving few input features. Such trees can be trained using the Tree Alternating Optimization (TAO) algorithm. Our approach is to train a sparse oblique tree that is as small and sparse as possible while achieving a good enough predictive accuracy, and then to inspect the weights in the tree decision nodes in order to establish a relationship between input features and classes. This reveals interesting patterns about the classifier and about the data itself. For example, we determine how small, specific subsets of features are used for specific classes (say, certain words for certain document topics), both globally or for a single input instance. The hierarchical structure of the tree also explains the common theme among a group of instances. We demonstrate this using the AG news dataset.**

***Index Terms*—interpretability, text classification, text mining, decision trees**

I. Introduction

Today machine learning models are widely deployed in many practical applications such as computer vision, text analysis, finance, medicine, and many others. However, despite many attempts, an aspect that has not been satisfactorily solved is the ability to interpret how a model predicts or provides any insights about the dataset. Deep neural networks are a classic example of this. They are highly accurate predictive models but lacks reasoning for their decisions [1], [2]. In the past few years, a lot of effort has been made in this direction, like providing an explanation for a given prediction [3]–[5] or what information is encoded by the model's parameters [6]–[10]. However, these methods fail to provide global explanations such as how the model differentiates between classes or subgroups within a class.

Ideally, we like to have a model that has high accuracy and a high degree of interpretability. This paper focuses on one such model, which is sparse oblique decision trees. We show that we can achieve very high accuracy using sparse oblique decision trees and provide interpretable predictions with extensive insights about the dataset itself. We demonstrate our approach effectively on text classification tasks.

Text classification on raw text data is difficult for many reasons. Different text instances have different lengths, and text can contain numbers or other symbols, making data handling difficult. So, over the years, several techniques have been proposed to convert raw text data into meaningful features, such as bag-of-words, bag-of-phrases, WordNet [11], and word-embedding [12], [13].

However, the extracted features do not convey much information about the dataset. This is because either the features are of large dimension (bag-of-words), or they are already non-interpretable (word-embedding [14]). Feature selection methods (Chi-Squared [15], Mutual Information [16], and Kolmogorov-Smirnov statistic [17]) help to reduce the dimensionality but still leaves a lot of questions unanswered. For instance, in the text classification task: we do not know how classes are arranged in the feature space, or if subgroups exist in a single class, and if they exist, what set of features make it happen; what common concept is represented by a single subgroup; or do the features we have selected are optimal for a given class, a subgroup of a class, or even for a single instance?

We address these issues by using sparse oblique decision trees as a tool to understand the relationship between the selected features and the dataset. For decades, decision trees have been widely recognized as highly interpretable models, but they have been mostly restricted to only axis-aligned trees [18], [19]. This is due to the lack of an algorithm that can train highly accurate oblique trees. Also, axis-aligned trees are only interpretable when the dataset is small. Unfortunately, as the input dimension grows, axis-aligned trees become massive and less accurate and no longer remain interpretable (fig. 4 and table: I). Here, we capitalize on a recently proposed *Tree Alternating Optimization (TAO)* algorithm [20], [21], which can learn far more accurate trees that remain small and very interpretable. Unlike axis-aligned trees that operate only on a single feature at each node, the oblique tree operates on a small, learnable subset of features. Secondly, unlike traditional tree algorithms such as CART [22] or C4.5 [23], TAO monotonically decreases the objective function containing classification error and sparsity penalty on the weights

of each node. It has been shown to outperform existing tree algorithms by a large margin [24], and to improve decision forests [25]–[30]. Counterfactual explanations can also be solved exactly for these trees [31]–[33]. Finally, the stronger predictive power of sparse oblique decision trees together with their interpretability and fast inference makes them useful for other uses, such as in understanding deep neural networks [34], [35] or compressing deep neural networks [36].

Also, soft trees, such as a hierarchical mixture of experts [37], can achieve good accuracy as they are differentiable and can be trained effectively using EM algorithm or gradient descent. However, unlike sparse oblique trees, they are not interpretable. Since the decisions are stochastic, every instance has to follow every single path to every leaf. This means a given instance reaches every leaf with different probabilities, which in many cases means that multiple leaves have a different proportion of probabilities. Therefore this makes interpreting such trees very difficult since one has to look at a large number of nodes that depend on a single instance.

Next, we discuss the TAO algorithm briefly, explain our approach to interpret the NLP datasets, describe the dataset used, and finally demonstrate our approach in the text classification task for the AG's news dataset.

II. Review: learning Sparse Oblique Trees with Tree Alternating Optimization (TAO)

Decision trees have been widely used in practice for decades. However, as any textbook will show [38], the type of trees that are almost universally used are axis-aligned, and the type of algorithm used to learn them is CART [22] or C4.5 [23]. Axis-aligned trees test a single feature at each decision node (e.g. "if $x_7 \geq 0.8$ go left else go right"). The CART or C4.5 training procedures grow the tree structure by recursively partitioning the training set, at each step greedily selecting the feature and threshold value by using a purity criterion (such as the Gini index or entropy). This has two important disadvantages. The first is that these algorithms are quite suboptimal. Their greedy nature means that wrong parameter choices early in the tree affect the choices made further down the tree. Also, those algorithms do not optimize a well-defined loss function jointly over all the tree parameters, instead using a proxy cost (purity) at each node in isolation. The second disadvantage is that the family of axis-aligned trees is very restrictive as a predictive model. This is because the discrimination boundaries it imposes are made of axis-aligned boxes, which works badly with correlated, high-dimensional feature vectors, leading to large trees that are hard to interpret and which have low predictive accuracy. Also, a binary axis-aligned tree having L leaves has $L-1$ decision nodes and so it uses, globally throughout the tree, at most $L-1$ input features. For a given input instance, the number of features examined is far smaller, upper-bounded by the depth of the root-leaf path it follows. For high-dimensional data, such as documents encoded as bag-of-words, the number of features D is far larger than the number of leaves or the depth of the tree. Hence, an axis-aligned tree is by definition limited to examining a small subset of the input features, which will drastically hurt its predictive accuracy.

Why is it so difficult to train decision trees as opposed to, say, neural nets or kernel machines? The reason is that the tree predictive function is piecewise constant, and hence non-differentiable with respect to the input features or the parameters. This poses a difficult optimization problem to which gradient-based techniques are not directly applicable. However, a recently proposed algorithm, *Tree Alternating Optimization (TAO)* [20], [21], is able to learn decision trees so that they optimize an objective function defined by a specific loss function and a regularization term. Besides, TAO can learn not just axis-aligned trees, but trees of more general form, in particular oblique trees. Next, we briefly review TAO.

TAO works very differently from CART and much more like a regular machine-learning optimization algorithm, but instead of gradients (which do not apply) it uses alternating optimization over groups of nodes of a fixed tree structure. This results in a monotonic decrease of the objective function over all the tree parameters and convergence to a local optimum. Consider a K-class classification problem with a training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{1,\dots,K\}$ of D-dimensional instances and labels. Let $T(\mathbf{x};\boldsymbol{\Theta})$ be a binary decision tree which produces a prediction for each input $\mathbf{x}$ by routing $\mathbf{x}$ from the root to exactly one leaf and applying a predictor function at that leaf. Each node (both decision and a leaf) has learnable parameters $\boldsymbol{\theta}_i$ and the total set of parameters of a tree is $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i\in\mathcal{N}}$, where $\mathcal{N} = \mathcal{D} \cup \mathcal{L}$ is the set of nodes (decision nodes and leaves). Specifically for a sparse oblique tree:

- Each decision node i has a decision function $f_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \mathcal{N}$ given by a *(sparse) hyperplane*: "go to the right child if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$, else go to the left child", with parameters $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$.
- Each leaf i has as parameter the label it predicts $\theta_i \in \{1,\dots,K\}$.

TAO assumes a fixed tree structure (say, complete of depth Δ) and initial node parameters, and optimizes the following objective function:

$$E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} L(y_n, T(\mathbf{x}_n;\boldsymbol{\Theta})) + \alpha \sum_{i\in\mathcal{L}} \|\boldsymbol{\theta}_i\|_1 \qquad (1)$$

where $L(\cdot,\cdot)$ is the 0/1 loss, and the ℓ_1 penalty over the weight vectors of the decision nodes promotes sparsity, via a hyperparameter $\alpha \geq 0$.

TAO is based on two theorems. First, eq. (1) *separates over any subset of non-descendant nodes* (e.g. all the nodes at the same depth); this follows from the fact that the tree makes hard decisions. All such nodes may be optimized in parallel. Second, optimizing over the parameters of a single node i simplifies to a well-defined *reduced problem* over the instances that currently reach node i (the *reduced set* $\mathcal{R}_i \subset \{1,\dots,N\}$). The form of the reduced problem depends on the type of node:

input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{1, \dots, K\}$
initial tree T (complete of depth Δ with random parameters)
repeat
  for $i \in$ nodes of T, visited in reverse BFS
    if i is a leaf **then**
      $y_i \leftarrow$ majority label of the training points that reach i
    else
      $\boldsymbol{\theta}_i \leftarrow$ minimizer of the reduced problem, eq. (2)
until stop
postprocess T: remove dead branches & pure subtrees
return T

Fig. 1. Pseudocode for the tree alternating optimization (TAO) algorithm to learn a sparse oblique tree (having a sparse hyperplane split at each decision node and a constant label at each leaf). Visiting each node in reverse breadth-first search (BFS) order means scanning depths from depth(T) down to 1, and processing at each depth (in parallel, if so desired) all nodes at that depth. "stop" occurs when either the objective function decreases less than a set tolerance or the number of iterations reaches a set limit.

- For a decision node $i \in \mathcal{D}$, it is a *weighted 0/1 loss binary classification problem*, where the two classes correspond to the left and right child, which are the only possible outcomes for an instance. Class $\texttt{left}_i$ ($\texttt{right}_i$) incurs a loss (weight) given by the prediction of the leaf reached from the left (right) child's subtree. Thus, each instance is assigned as *pseudolabel* the child with lower loss. The reduced problem takes the form (where $\overline{L}$ and $\overline{y}_n$ are the said loss and pseudolabel, respectively):

$$E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} \overline{L}(\overline{y}_n, f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha \left\|\boldsymbol{\theta}_i\right\|_1. \quad (2)$$

  This problem is NP-hard but can be well approximated with a convex surrogate; we use ℓ_1-regularized logistic regression where each instance is weighted by the loss difference between the winner child and the other child, and solve it using LIBLINEAR [39].
- For a leaf node $i \in \mathcal{L}$, the reduced problem consists of optimizing the original loss but over the leaf label θ_i on its reduced set:

$$E_i(\theta_i) = \sum_{n \in \mathcal{R}_i} L(y_n, \theta_i) \quad (3)$$

  whose optimal solution is found by setting θ_i to the majority class label in the reduced set.

The resulting algorithm visits nodes in reverse breadth-first search order and is shown in figure 1. Essentially, each iteration trains all nodes at the same depth (in parallel) from the leaves to the root, by solving either an ℓ_1-regularized logistic regression at each decision node, or setting each leaf to the majority vote. Importantly, the ℓ_1 penalty on the decision nodes' weight vectors can force some of them to become zero ($\mathbf{w}_i = \mathbf{0}$). This makes such nodes redundant and they can be pruned at the end, reducing the size of the tree.

The computational complexity of TAO to train a sparse oblique classification tree can be characterized as follows. Consider a complete tree of depth Δ and assume training each decision node (ℓ_1-regularized logistic regression) is linear on the sample size. Then, training all the decision nodes at the same depth is approximately constant and equal to training one logistic regression on the whole training set. The cost of training the leaves is negligible compared to that. Thus, the total sequential cost of one iteration is approximately equal to that of one Δ logistic regressions on the whole dataset. As noted above, all the nodes at the same depth can be trained in parallel, which can considerably reduce the training time.

TAO has been shown to improve consistently and by a wide margin the predictive accuracy for a wide range of tree-based models, beyond classification trees [20], [24], such as regression trees [26], tree ensembles [25], hybrid models [40], etc. The resulting sparse oblique trees, in addition to being highly accurate, are quite small in size and use few weights in the decision nodes, which makes them highly interpretable. We capitalize on this in this paper in the context of NLP datasets and tasks, as shown next.

III. Dataset description

AG is a collection of news articles, and these articles have been collected from more than 2 000 news sources. In this work, we use AG's news topic classification dataset [41][1]. It has been categorized into 4 categories: world, sports, business, and science/technology. There are 60 000 training and 7 600 test instances. We first transform the data into TF-IDF features [42], which creates a vocabulary size of 53 740, out of those, we choose 3 000 features using Chi-Squared [15].

IV. Methodology

Our approach is as follows:

1) Train a sparse oblique tree using TAO [20], [21] and choose the sparsity parameter such that the tree is as sparse as possible, while remaining sufficiently accurate.
2) Use the weights of the tree to establish a relationship between features and the classes.

For input $\mathbf{x} \in \mathbb{R}^d$ we define decision rule at a decision node i as follows: "if $\mathbf{w}_i^T \mathbf{x} + b_i \geq 0$ then go to right child else go to left child", where $\mathbf{w}_i \in \mathbb{R}^d$ is the weight vector and $\mathbf{b}_i \in \mathbb{R}$ is

[1] https://huggingface.co/datasets/ag_news

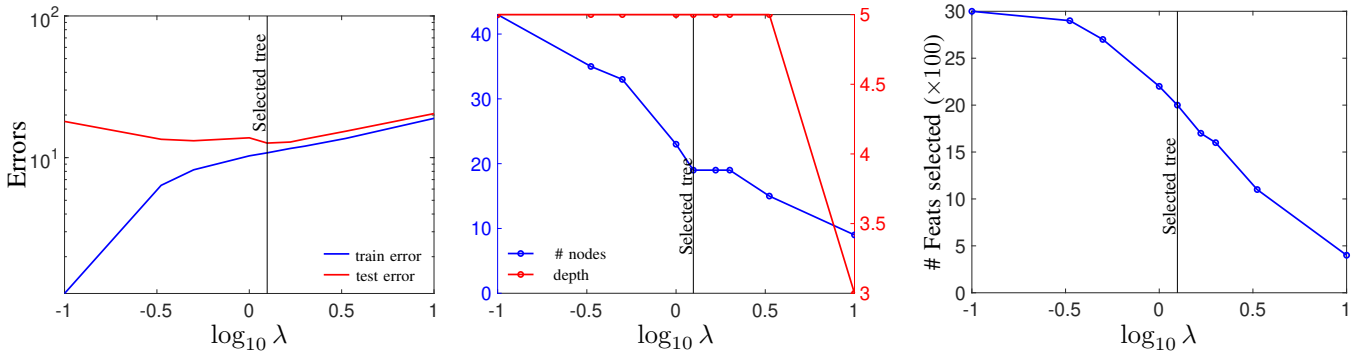

Fig. 2. Classification error (training and test) and number of nodes and of features selected by the trees as a function of λ for AG's news. The vertical line indicates the tree we selected as mimick ($\lambda = 1.25$).

the bias. Now, to associate the features of $\mathbf{x}$ with the weights of a node i, we apply the following operations:

Write $\mathbf{w}$ and $\mathbf{x}$ as $\mathbf{w} = (\mathbf{w}_0\ \mathbf{w}_-\ \mathbf{w}_+)$ and $\mathbf{x} = (\mathbf{x}_0\ \mathbf{x}_-\ \mathbf{x}_+)$, where $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{w}_- < \mathbf{0}$ and $\mathbf{w}_+ > \mathbf{0}$ contain the zero, negative and positive weights in $\mathbf{w}$, and $\mathbf{x} \geq \mathbf{0}$ [2] is arranged accordingly. Call $\mathcal{S}_0$, $\mathcal{S}_-$ and $\mathcal{S}_+$ the corresponding sets of indices in $\mathbf{w}$. Then $\mathbf{w}^T\mathbf{x}+b = \mathbf{w}_-^T\mathbf{x}_- + \mathbf{w}_+^T\mathbf{x}_+$ with $\mathbf{w}_-^T\mathbf{x}_- \leq 0$ and $\mathbf{w}_+^T\mathbf{x}_+ \geq 0$. So if $\mathbf{x}_- = \mathbf{0}$ then $\mathbf{w}^T\mathbf{x} + b \geq 0$ and $\mathbf{x}$ would go to the right child and if $\mathbf{x}_+ = \mathbf{0}$ then $\mathbf{w}^T\mathbf{x} + b \leq 0$ and $\mathbf{x}$ would go to the left child. Next, if $\mathbf{x}$ goes to left, we represent the feature selected as a binary vector $\mu^- \in \{0,1\}^d$ containing ones only at $\mathcal{S}_-$. Similarly, if $\mathbf{x}$ goes to right, binary vector μ^+ is one only at $\mathcal{S}_+$. We call μ^+ and μ^- the NODE-FEATURES, where location of one represents features selected by $\mathbf{w}$.

We use NODE-FEATURES in the following way to interpret the dataset:

1) The NODE-FEATURES of a decision node represents the features that are responsible for nodes in left and right subtree. By examining them, we can understand what features separates a group of classes or subgroup of a given class.
2) We can find features specific to class k as follows: find the path from the root to the leaf of class k. For each node i, find NODE-FEATURES along the path, and in the end take logical OR of the NODE-FEATURES. If there are more then one leaf for class k, take union of all the features selected.
3) We can find features specific to a given input $\mathbf{x}$ as follows: repeat the same process as above, but only for the leaf containing $\mathbf{x}$. Next, among all selected features, keep only those which have non-zero value in the TF-IDF representation of $\mathbf{x}$.

Although features selected by NODE-FEATURES are in TF-IDF format, we can map the TF-IDF element to its corresponding word, and thus make the features explainable.

V. EXPERIMENT

As described above, we first transform the AG's news dataset into TFIDF features, where each data point is in $\mathbb{R}^{3000}$

[2]TF-IDF features are non-negative.

TABLE I
TRAINING AND TEST ERRORS, AND PARAMETER FOR CART AND TAO ON AG'S NEWS DATASET

Model	Training Error	Test Error	Height	# Nodes
TAO	10.81	12.67	5	19
CART	34.13	40.36	30	2679

space. We used an initial tree structure for TAO of depth 6 (total 127 nodes) and random initial values for the weights at the nodes. The decision nodes are hyperplanes, and each leaf contains a single class label. We constructed a collection of trees over a range of the sparsity hyperparameter $\lambda \in [0, \infty)$. In fig. 2 we show the training/test errors as a function of sparsity parameters. From there, we pick the tree with the best test accuracy ($\lambda = 1.25$). We have also tried the initial tree structure with different depths, but the initial tree tree with depth 6 performs best in test accuracy. In fig. 3, we show the weight vector (left) and histogram of instances (right) at each node of the tree. In table I, we show a comparison between TAO and CART in terms of accuracy and the number of parameters. As discussed earlier, CART only works if the dataset is small and easy. Otherwise, it is very difficult to interpret, as shown in fig. 4.

A. Data interpretation using tree

1) Position of sports class in feature space: As described above, each non-zero element in the weight vector represents a word in the vocabulary. Just by looking at the tree, it is clear to see how the classes are distributed in the vocabulary space. The class sports is easier to separate from other classes, as the most features selected by the root for the left child (μ_1^-) belong to class sports. As shown in the table: II, the tree needs only 579 out of 2692 features.

2) 3 subgroups of business class: leaf # 13: On the other hand, the class business is much harder to separate. The tree requires 1 618 out of 2 817 features. However, the tree categorized the instances into three groups (leaf number 18, 17, and 13). Decision node number 3, divides the class into three subgroups. It sends leaf 13 to the right subtree and rest to the left subtree. Among all the features

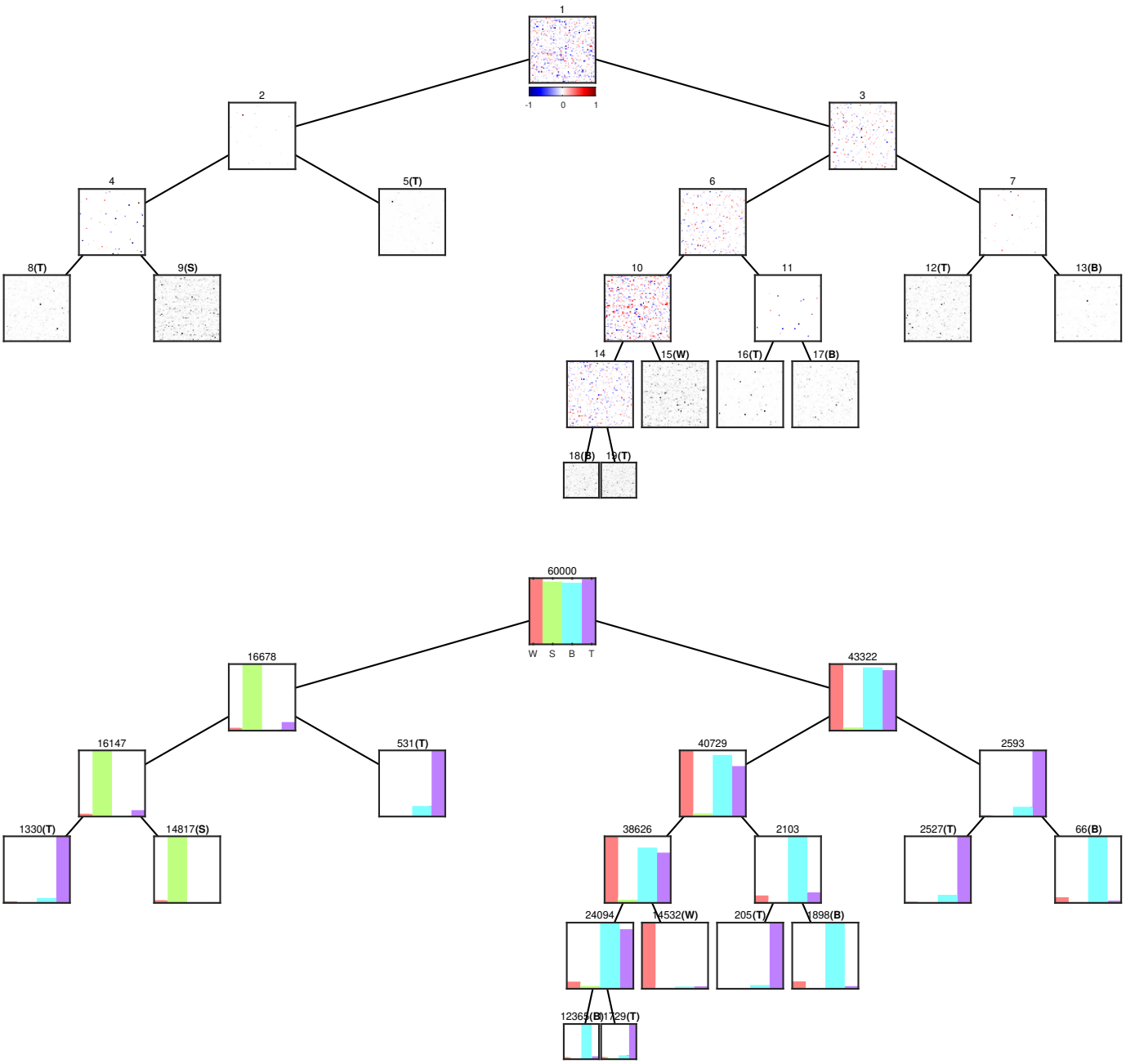

Fig. 3. Tree selected to interpret the AG's news dataset ($\lambda = 1.25$). *Top:* weight vector at each decision node and average of training instances at each leaf; we show the node index and, for leaves, also their label. We plot both the weight vector and the average, of dimension 3 000, as a 55×55 square (the last pixels are unused), with features in the original order in TF-IDF representation. Weight vectors are colored according to their sign and magnitude (positive, negative and zero values are blue, red, and white, respectively), and averages as greyscale. *Bottom:* class histograms; we show the number of training instances reaching the node and, for leaves, their label. W, S, B and T represents world, sports, business, and science/technology class respectively. You may need to zoom in the plot.

associated with right subtree (μ_3^+) is `halliburton`; in fact, throughout the tree, `halliburton` appears only twice, and that too, in the path of the leaf number 13. When we look at the inputs in this leaf, most of them are related to the "Halliburton" company. Some of the features associated with this leaf and μ_3^+ are: {`oil, filed, Iraq, united, president, discovered`}; which makes sense as, `halliburton` is the name of a multinational oil company.

3) 3 subgroups of business class: leaf # 17: Next, to understand the difference between the class subgroup at leaf number 17 and 18, we focus on decision node number 6. This node sends leaf number 17 to the right subtree and leaf number 18 to the left subtree. Some features related to μ_6^+ are: {`market, CEO, chairman, debt, deficit, dollar, IPO, monetary, fiscal, capital, stock`}. This indicates that the leaf number 17 contains the business news that

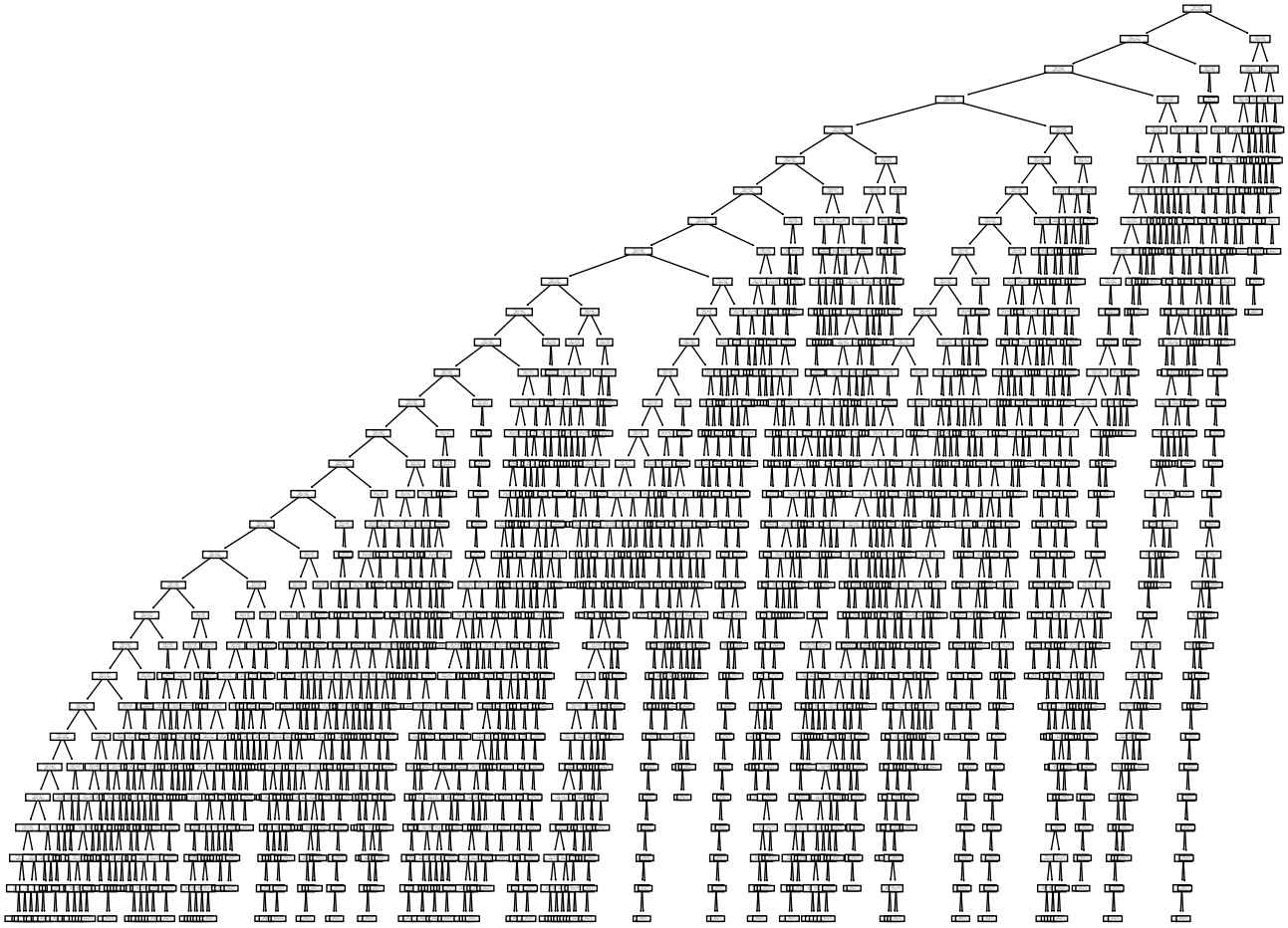

Fig. 4. CART tree trained on AG's news dataset.

TABLE II
NUMBER OF FEATURE SELECTED FOR EACH CLASS.

Model	Class				Total
	World	Sports	Business	Sci/Tech	
Original # non zero features	2902	2692	2817	2844	3000
# features selected by TAO	958	579	1175	1618	1969
# features selected by LASSO			-		2015

is specific to finance. Indeed, when we look at instances in leaf number 17, they are mostly finance-based news articles like:

"*NEW YORK - Newly optimistic investors sent stocks sharply higher Friday, propelling the Dow Jones industrials 112 points higher, as new economic data showed strength in manufacturing and the departure of PeopleSoft Inc.'s chief executive raised hopes for a merger in the tech sector. The major indexes closed the week with respectable gains.*"

4) 3 subgroups of business class: leaf #18: On the other hand, features related to μ_6^- are {`soldier, killing, piracy, coup, attack, threatening, sanction, UN`}. This is because the class subgroup contains business news articles that are related to world politics. That is why this class subgroup appears in the same subtree as the class world news, and have articles like this:

"*NEW YORK (Reuters) - Oil prices ended above $50 a barrel on Friday amid concern over possible fighting between rebels and the military in Nigeria's oil-rich delta region.*"

This shows how hierarchical structure of the tree allows to interpret the data in a better way.

B. Interpreting individual instance

In table: III we show an example of selecting features for a single instance. As shown in the second row, the features selected by the tree already removes a lot of words like { `Norwegian, famed, again`}. Next, when we look at the class sci/tech specific features, the tree keeps only tech-related features, such as {`dvd, wireless, apple, hacker, software`}. This removes {`apparently, struck, breaking` }, which on its own, do not belong to sci/tech class. Finally, when we apply instance-based feature extraction, we see that for the given instance, we do not need

words like { ap, time, hacker}, which are tech-related but not for the given instance.

TABLE III
FEATURE SELECTION FOR AN INSTANCE OF THE CLASS SCI/TECH

Original text	AP - *The Norwegian hacker famed for developing DVD encryption-cracking software has apparently struck again #151; this time breaking the locks on Apple Computer Inc.'s wireless music streaming technology.*
Common features between classes	apparently, dvd, wireless, struck, inc, technology, developing, time, apple, software, computer, breaking, music, hacker, ap
Class specific features	dvd, wireless, inc, technology, developing, time, apple, software, computer, music, hacker, ap
Instance Specific features	dvd, wireless, inc, technology, developing, apple, software, computer, music

C. Selected features and word-frequencies

One can argue that we can select the relevant features (words) for a given class by considering only word frequencies. That is, for a given class, we simply count how many times a word appears in the class specific training instances (text data before TF-IDF conversion). If a word frequently appears in the articles related to a specific class, it should be relevant to that class. However, this is not always true. For example, in the case of class world news, there are several high-frequency words (table IV) that the tree does not select. These are mostly the words that are not distinctively related to the world news. On the other hand, there are some low-frequency words (table IV) that the tree selects as necessary for the world news class. Although these words appear very rarely in the dataset (training instances labeled as world news class), when they appear in a particular instance, they are essential for that instance. That is why the tree selects these words.

TABLE IV
WORDS SELECTED FROM THE ARTICLES THAT BELONG TO WORLD NEWS CLASS. EACH WORD IS REPRESENTED ALONG WITH ITS FREQUENCY REPRESENTED AS WORD(FREQUENCY).

	word(frequency)
high frequency words not selected by the tree	year(979), first(754), last(596), quot(548), car(469), time (418), near(345), news(336), next(321), night(319)
low frequency words selected by the tree	crown(16), sweden(20), hollywood(22), famous(24), austria(32), blood(34), earthquake(35), christian(37), shooting(38), poverty(39)

VI. CONCLUSION

In this work, we introduce sparse oblique trees as an accurate, yet interpretable tool for text-classification. We have also shown how sparse oblique trees can uncover the underlying patterns in the text data. Thus, it provides a simple yet effective method to interpret the dataset and understand how different features interact. We have also introduced a hierarchical feature selection method that does feature selection per class and per instance. This provides a way to use optimal number features based on the number of instances.

Acknowledgments. Work funded in part by NSF award IIS–2007147.

REFERENCES

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, p. 93, May 2018.

[2] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Machine Intelligence*, vol. 1, no. 5, pp. 206–215, May 2019.

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2016)*, San Francisco, CA, Aug. 13–17 2016, pp. 1135–1144.

[4] W. J. Murdoch, P. J. Liu, and B. Yu, "Beyond word importance: Contextual decomposition to extract interactions from LSTMs," in *Proc. of the 6th Int. Conf. Learning Representations (ICLR 2018)*, Vancouver, Canada, Apr. 30 – May 3 2018.

[5] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. of the 2016 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, Jun. 26 – Jul. 1 2016.

[6] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," Université de Montréal, Tech. Rep. 1341, Jun. 2009.

[7] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proc. of the 2nd Int. Conf. Learning Representations (ICLR 2014)*, Banff, Canada, Apr. 14–16 2014.

[8] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems (NIPS)*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. MIT Press, Cambridge, MA, 2016, pp. 3387–3395.

[9] S. S. Hada and M. Á. Carreira-Perpiñán, "Sampling the "inverse set" of a neuron: An approach to understanding neural nets," Sep. 27 2019, arXiv:1910.04857.

[10] ——, "Sampling the "inverse set" of a neuron," in *IEEE Int. Conf. Image Processing (ICIP 2021)*, Online, Sep. 19–22 2021, pp. 3712–3716.

[11] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, pp. 39–41, 1995.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems (NIPS)*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. MIT Press, Cambridge, MA, 2013, pp. 3111–3119.

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," Jul. 16 2016, arXiv:1607.01759.

[14] Z. C. Lipton, "The mythos of model interpretability," *Comm. ACM*, vol. 81, no. 10, pp. 36–43, Oct. 2018.

[15] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Machine Learning Research*, vol. 3, pp. 1289–1305, Mar. 2003.

[16] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[17] D. J. Dittman, T. M. Khoshgoftaar, R. Wald, and J. V. Hulse, "Comparative analysis of dna microarray data through the use of feature selection techniques," in *9th Int. Conf. Machine Learning and Applications (ICMLA)*, Washington DC, Dec. 12–14 2010, pp. 147–152.

[18] C. Apte, F. Damerau, and S. Weiss, *Text mining with decision rules and decision trees*. Citeseer, 1998.

[19] A. A. Nasseri, A. Tucker, and S. de Cesare, "Quantifying StockTwits semantic terms' trading behavior in financial markets: An effective application of decision tree algorithms," *Expert systems with applications*, vol. 42, no. 23, pp. 9192–9210, 2015.

[20] M. Á. Carreira-Perpiñán and P. Tavallali, "Alternating optimization of decision trees, with application to learning sparse oblique trees," in *Advances in Neural Information Processing Systems (NEURIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. MIT Press, Cambridge, MA, 2018, pp. 1211–1221.

[21] M. Á. Carreira-Perpiñán, "The Tree Alternating Optimization (TAO) algorithm: A new way to learn decision trees and tree-based models," 2021, arXiv.

[22] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, Calif.: Wadsworth, 1984.

[23] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[24] A. Zharmagambetov, S. S. Hada, M. Gabidolla, and M. Á. Carreira-Perpiñán, "Non-greedy algorithms for decision tree optimization: An experimental comparison," in *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, Jul. 18–22 2021.

[25] M. Á. Carreira-Perpiñán and A. Zharmagambetov, "Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting," in *Proc. of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020)*, Seattle, WA, Oct. 19–20 2020, pp. 35–46.

[26] A. Zharmagambetov and M. Á. Carreira-Perpiñán, "Smaller, more accurate regression forests using tree alternating optimization," in *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, H. Daumé III and A. Singh, Eds., Online, Jul. 13–18 2020, pp. 11 398–11 408.

[27] A. Zharmagambetov, M. Gabidolla, and M. Á. Carreira-Perpiñán, "Improved boosted regression forests through non-greedy tree optimization," in *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, Jul. 18–22 2021.

[28] ——, "Improved multiclass AdaBoost for image classification: The role of tree optimization," in *IEEE Int. Conf. Image Processing (ICIP 2021)*, Online, Sep. 19–22 2021, pp. 424–428.

[29] M. Gabidolla and M. Á. Carreira-Perpiñán, "Pushing the envelope of gradient boosting forests via globally-optimized oblique trees," in *Proc. of the 2022 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'22)*, New Orleans, LA, Jun. 19–24 2022.

[30] M. Gabidolla, A. Zharmagambetov, and M. Á. Carreira-Perpiñán, "Improved multiclass AdaBoost using sparse oblique decision trees," in *Int. J. Conf. Neural Networks (IJCNN'22)*, Padua, Italy, Jul. 18–22 2022.

[31] M. Á. Carreira-Perpiñán and S. S. Hada, "Counterfactual explanations for oblique decision trees: Exact, efficient algorithms," in *Proc. of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, Online, Feb. 2–9 2021, pp. 6903–6911.

[32] ——, "Counterfactual explanations for oblique decision trees: Exact, efficient algorithms," Mar. 1 2021, arXiv:2103.01096.

[33] S. S. Hada and M. Á. Carreira-Perpiñán, "Exploring counterfactual explanations for classification and regression trees," in *ECML PKDD 3rd Int. Workshop and Tutorial on eXplainable Knowledge Discovery in Data Mining (XKDD 2021)*, 2021, pp. 489–504.

[34] S. S. Hada, M. Á. Carreira-Perpiñán, and A. Zharmagambetov, "Sparse oblique decision trees: A tool to understand and manipulate neural net features," Apr. 7 2021, arXiv:2104.02922.

[35] ——, "Understanding and manipulating neural net features using sparse oblique classification trees," in *IEEE Int. Conf. Image Processing (ICIP 2021)*, Online, Sep. 19–22 2021, pp. 3707–3711.

[36] Y. Idelbayev, A. Zharmagambetov, M. Gabidolla, and M. Á. Carreira-Perpiñán, "Faster neural net inference via forests of sparse oblique decision trees," 2022, arXiv.

[37] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, Mar. 1994.

[38] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning—Data Mining, Inference and Prediction*, 2nd ed., ser. Springer Series in Statistics. Springer-Verlag, 2009.

[39] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, vol. 9, pp. 1871–1874, Aug. 2008.

[40] A. Zharmagambetov and M. Á. Carreira-Perpiñán, "Learning a tree of neural nets," in *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, Toronto, Canada, Jun. 6–11 2021, pp. 3140–3144.

[41] X. Zhang, J. Zhao, and Y. LeCun, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. MIT Press, Cambridge, MA, 2012, pp. 649–657.

[42] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.