DISK-CSV: Distilling Interpretable Semantic Knowledge with a Class Semantic Vector

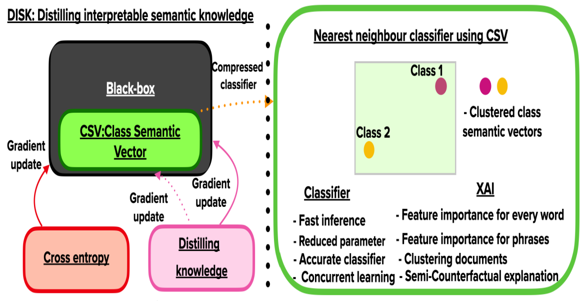
Housam Khalifa Bashier Babiker [1], **Mi-Young Kim**[2], **Randy Goebel**[1]
Alberta Machine Intelligence Institute
[1] Department of Computing Science, University of Alberta
[2] Department of Science, Augustana Faculty, University of Alberta
{khalifab, miyoung2, rgoebel}@ulaberta.ca

Abstract

Neural networks (NN) applied to natural language processing (NLP) are becoming deeper and more complex, making them increasingly difficult to understand and interpret. Even in applications of limited scope on fixed data, the creation of these complex "black-boxes" creates substantial challenges for debugging, understanding, and generalization. But rapid development in this field has now lead to building more straightforward and interpretable models. We propose a new technique (DISK-CSV) to distill knowledge concurrently from any neural network architecture for text classification, captured as a lightweight interpretable/explainable classifier. Across multiple datasets, our approach achieves better performance than the target black-box. In addition, our approach provides better explanations than existing techniques.

1 Introduction

Deep Neural Networks (DNNs) are popular in many applications, including computer vision and natural language processing. However, two major factors still require attention: (1) understanding the classifier's prediction for the end-users to develop trust in the model, and to enable machine learning engineers to refine it. This is especially true for high-stakes domains such as clinical decision support. There has been attention on models that attempt to make a neural network explainable, for instance, (Sundararajan et al., 2017), and (Ribeiro et al., 2016), which create post-hoc explanations to support explainability. Secondly, (2) artificially high numbers of parameters make the inference time expensive.


DISK: Distilling interpretable semantic knowledge
Black-box
CSV:Class Semantic Vector
Compressed classifier
Gradient update
Gradient update
Gradient update
Cross entropy
Distilling knowledge
Nearest neighbour classifier using CSV
Class 1
Class 2
- Clustered class semantic vectors
Classifier
- Fast inference
- Reduced parameter
- Accurate classifier
- Concurrent learning
XAI
- Feature importance for every word
- Feature importance for phrases
- Clustering documents
- Semi-Counterfactual explanation
Figure 1: Our proposed model.

Neural networks tend to be deep, with millions of parameters. For example, GPT-2 (Radford et al., 2019) needs over 1.5 billion parameters. As a result, they are compute-intensive, thus making it difficult to deploy in real-world applications. We here propose a model-agnostic interpretable knowledge distillation method for neural network text classification.

As shown in Figure 1, we learn a class semantic vector for each output class, concurrently when training the black box. We then use the semantic vectors to create the nearest neighbor classifier (compressed interpretable/explainable classifier) from the black-box version. Knowledge distillation refers to the process of transferring the implicit knowledge learned by a teacher model to a student model (Liu and Matwin, 2018). Dark knowledge refers to the salient information hidden in the predicted probabilities for all classes, which are more informative than the predicted classes themselves. Our contributions can be summarized as follows:

- We propose a knowledge distillation method where dark knowledge can be learned concurrently by a student model, while building a black-box model.
- We propose an interpretable classifier, which provides a user explanation for predicting a single class label.

- We integrate a clustering technique within our interpretable classifier model.
- We provide an interactive explanation mode, where users can directly request a word or a phrase query and receive feedback.
- Our smaller model shows even better performance than the original black-box, with drastically reduced hyper-parameters. That smaller model can be deployed as an on-line service in real-time applications in resource-restricted devices.

2 Related Work

This work has connection with research on explainablity and model compression.

Explainability. Most of the existing explainable AI (XAI) techniques for Natural Language Processing text classification focus on assigning a score to each word in the document w.r.t. predicted class, typically using gradient-based or perturbation-based methods (Arras et al., 2017; Sundararajan et al., 2017; Shrikumar et al., 2017a; Bach et al., 2015). The most popular technique for model-agnostic explanation is LIME (Ribeiro et al., 2016), which focuses on creating an interpretable classifier by approximating it locally, with a linear model.

The main drawback of these methods is that those explanations are not faithful to the target model (Rudin, 2018). There are other methods, which focused on constructing a self-explainable network (Bastings et al., 2019) and (Lei et al., 2016). These techniques have limited explanations and thus do not explain phrases. Our work is different from post-hoc and self-explainable approaches as it attempts to learn an explainable smaller classifier concurrently with the target black-box model. Our explanations are also generated from the interpretable classifier itself, without extra calculation as in post-hoc techniques.

Model compression. A variety of research devoted their efforts to compressing large networks to accelerate inference, transfer, and storage. One of the earliest attempts focused on pruning unimportant weights (LeCun et al., 1990). Other methods focused on modifying devices to improve floating point operations (Tang et al., 2018). In contrast, some works focused on quantizing neural networks (Wu et al., 2018). Other investigations have focused on knowledge distillation, i.e., the ability to transfer the knowledge from a larger model to a smaller model (Ba and Caruana, 2014) and (Hinton et al., 2015). However, the main drawbacks of the methods mentioned above are that: (1) they only work with pre-trained networks, (2) the compressed models are still treated as black-box, and (3) the compression techniques require another training step or additional computation which complicates the process. In contrast, we concurrently transfer the knowledge from the black-box into a smaller interpretable model.

3 DISK-CSV

In a text classification task, an input sequence $x = \langle x_1, ..., x_l \rangle, x_i \in \mathbb{R}^d$, where l is the length of the input text and d is the vector dimension, is mapped to a distribution over class labels using a parameterized θ neural network model (e.g., a Long Short Term Memory network (LSTM), Transformer, etc.), which we denote as $\mathcal{F}(x; \theta)$. The output y is a vector of class probabilities, and the predicted class $\hat{y}$ is a categorical outcome, such as an entailment decision. In this work, we are interested in learning a simpler compressed nearest neighbor classifier (e.g., easy-to-explain its prediction) from any neural network model, but concurrently, while training the larger model. We refer to the large model (black-box) as $\mathcal{T}$ and the smaller interpretable/explainable model as $\mathcal{S}$.

We call our method **DISK-CSV** - **D**istilling **I**nterpretable **S**emantic **K**nowledge with a **C**lass **S**emantic **V**ector. In the next subsections we provide the following details for **DISK-CSV**: (a) how to distill knowledge from $\mathcal{T}$ into $\mathcal{S}$, (b) how to construct interpretable representations for $\mathcal{S}$, and (c) how to interact with the model to achieve better explainability (e.g., by clustering data, explaining essential phrases, and providing a semi-counterfactual explanation). Neural networks learn by optimizing a loss function to reflect the true objective of the end-user. For $\mathcal{S}$, our objective is to generalize in the same way as $\mathcal{T}$ and approximate an explanation for each prediction. To demonstrate our idea, we show how we can learn $\mathcal{S}$ concurrently with a long short-term memory network (LSTM) and then discuss how it can be generalized to different types of architectures for text classification. An LSTM network processes the input word by word, and at time-step t, the memory c_t and the hidden state h_t are updated. The last state $h_l \in \mathbb{R}^d$ is fed into a linear layer with parameters

$\boldsymbol{W} \in \mathbb{R}^{d \times k}$ which gives a probability distribution over k classes:

$$p(y|x) = \frac{\exp(h_l \cdot \boldsymbol{W})}{\sum_{i=1}^{k} \exp(h_l \cdot \boldsymbol{W})_k}, \tag{1}$$

The classifier uses cross-entropy loss to penalize miss-classification as $\mathcal{L}^{classifier} = -\frac{1}{k}\sum_{i=1}^{k} r_i \log(y_i)$, where $r \in \mathbb{R}^k$ is the one-hot represented ground truth and r_i is the target probability (0 or 1) for class i. The network's weights are updated via back-propagation when training the black-box. We intend to augment the neural nets that typically use embeddings to represent discrete variables (e.g., words) as continuous vectors. Words that have a similar context will have similar meanings. The simplest form of concurrent knowledge distillation is to transfer the knowledge from the embedding space of $\mathcal{T}$ into a k-**C**lass **S**emantic **V**ectors (**CSV**s) $v_i \in v$, where the dimension of each v_i is equal to the dimension of the embedding vector x_i, and k is the number of target classes. In other words, for each class label, we would like to learn a vector that captures the semantic information related to that class from the embedding layer.

These semantic vectors have the following properties: (1) Each vector v_i should capture/encode the semantics about the class i from the black-box; (2) These vectors are used by the nearest neighbor classifier for the prediction of the correct class label; (3) By using cosine similarity, we can compute the contribution of each word in the input with the corresponding v_i to the class i; (4) These vectors add another level of abstraction by explaining the feature importance of a phrase that expands a single word, and (5) The weights of the CSV are initialized in the same way we initialize the embedding layer and adjusted via back-propagation. We reformulate the optimization of $\mathcal{T}$ to update/adjust the weights of the **CSV**s as follows:

$$\mathcal{L} = \mathcal{L}^{classifier} + \lambda_1 \overbrace{\left(1 - \frac{\bar{x} \bullet tanh(v_{\hat{y}})}{\|\bar{x}\| \, \|tanh(v_{\hat{y}})\|}\right)}^{Semantics} + \lambda_2 \underbrace{\left(1 - \frac{h_l \bullet tanh(v_{\hat{y}})}{\|h_l\| \, \|tanh(v_{\hat{y}})\|}\right)}_{Hidden-knowledge} - \lambda_3 \left(\overbrace{D}^{Vectors-separation} \right) \tag{2}$$

where D is the pairwise Euclidean distance defined as $D = \frac{\sum_i \sum_j ((v_i - v_j)(v_i - v_j)^T)}{2}$, $\hat{y}$ is the index of the predicted class with the highest probability, and $\{\lambda_1, \lambda_2, \lambda_t r3\}$ are used to weight the importance of the terms. In what follows, we discuss the new terms added to the optimization problem.

Capturing semantics: The second term of Equation 2 is the second loss function we use, which attempts to encode the information of semantically consistent sentences in a single **CSV** $v_{\hat{y}}$. An obvious way to learn semantic information is to minimize the cosine distance between the average of the embedding $\bar{x}$ of the input sentence x and the predicted class semantic vector $v_{\hat{y}}$. This objective will ensure that the vector v_i captures the semantics of consistent inputs to encourage semantic consistency.

Hidden knowledge extraction: The last hidden state h_l in recurrent nets is typically used by the output layer as the feature vector to predict the class label. As a result, the salient information learned by $\mathcal{T}$ is encoded in this feature vector. To distill this knowledge and enrich the representation of v_i so that $\mathcal{S}$ generalizes well, we again minimize the cosine distance between the class semantic vector v_i and the last hidden state h_l in the third loss function, which is the third term in Equation 2. This objective allows the model $\mathcal{S}$ to generalize similarly to the black-box $\mathcal{T}$. The only constraint here is that the dimension of h_l must be the same as that of x_i so that we can minimize the cosine distance, i.e., $h_l \in \mathbb{R}^d$.

Vector-separation: Our ultimate goal is to create a simple interpretable nearest neighbor classifier $\mathcal{S}$ from the black-box. Therefore, we want to make sure that the **CSV**s are well separated from each other so that the cosine distance is maximum between them. To address this problem, we maximize the pairwise Euclidean distance between these vectors using the fourth term in Equation 2.

3.1 The smaller interpretable classifier $\mathcal{S}$ based on CSV

Our smaller model $\mathcal{S}$ is the nearest neighbor classifier, which relies mainly on the semantic information encoded in the vectors v learned via back-propagation when training $\mathcal{T}$. The model $\mathcal{S}$ takes the input sentence and computes the average $\bar{x}$ of the input embedding. Then we compute the cosine distance between $\bar{x}$ and each v_i. Finally, the target class is decided as the index i of the v_i with the lowest cosine distance. Besides, this classifier is interpretable, i.e., we can understand the mechanism of making a prediction. It can also be easily transferred or stored. The smaller model extracts the semantics from the larger model concurrently when training the black-box model. The algorithm is summarized in Figure 2.

```
Input: Sentence embedding $x$, CSVs $v$, $Dist = 5$
Output: Predicted class $\hat{y}$
1: $\bar{x}$ = average˙emebdding($x$)
2: for Each vector $v_i \in v$ do
3:     $tmpDist$ = CosineDistance $(v_i, \bar{x})$
4:     if $tmpDist < Dist$ then
5:         $\hat{y} \leftarrow i$
6:         $Dist \leftarrow tmpDist$
7:     end if
8: end for
```

Figure 2: Our smaller model $\mathcal{S}$ using **CSV**.

3.2 Explainability

Our model $\mathcal{S}$ provides four levels of explanations for text classification: (1) Word feature importance, (2) Document clustering, (3) End-user interaction through phrase feature importance, and (4) Semi-counterfactual explanation.

- **Word feature importance**: To understand the contribution of each word w.r.t. predicted class $\hat{y}$, we rely on the semantic similarity between each x_i and the nearest class vector $v_{\hat{y}}$. A word with high semantic similarity with the predicted $v_{\hat{y}}$ will have a high contribution to the predicted class. To understand the semantic contribution of each word in the input, we calculate the semantic similarity of every word to $v_{\hat{y}}$ using cosine similarity between the embedding vector of the input word and $v_{\hat{y}}$.

- **Document clustering**: Every text instance is clustered around its class semantic vector by computing the mean (x-axis) and the standard deviation (y-axis) of the elements in the vector $(\frac{\bar{x}+tanh(v_{\hat{y}})}{2})$ for each $\bar{x}$, where $v_{\hat{y}}$ is the nearest **CSV** to the input document. We found that the 2-D points (mean, standard deviation) of the elements in the vector $(\frac{\bar{x}+tanh(v_{\hat{y}})}{2})$ for the instances belonging to a specific class are close to each other and far from those of the instances belonging to other classes. The merit is that we do not need to use a clustering algorithm.

- **End-user interaction through phrase feature importance**: Word feature importance is sometimes not enough to explain a model's prediction. The end-user might also be interested in querying the classifier to answer different types of questions. For example, in the situation where the model shows the feature importance (in sentiment classification) of each individual word "not," "too," and "bad," an end-user might also be interested in the importance of the phrase "not too bad," which cannot be calculated just by merging the three different feature importance values. Our approach is capable of giving feedback to the user's query about a phrase. To obtain the feature importance for a phrase, we average the embedding vectors of the words in the phrases and then compute the cosine similarity w.r.t. the predicted **CSV** $v_{\hat{y}}$.

- **Semi-counterfactual explanation**: Our approach is also capable of providing a semi-counterfactual explanation, i.e., explaining a semi-casual situation (i.e., what kind of features prevent the classifier from changing the prediction to another class). We can provide a feature importance value w.r.t. non-predicted classes by calculating the cosine similarity between the embedding vector of each word/phrase and the class semantic vector of a non-predicted class). Through this semi-counterfactual explanation, the user can reason that "if the feature X had not occurred, the class prediction would have been changed."

3.3 Generalizing to other models

Our method can be adapted to a variety of architectures such as Bi-LSTM, GRU, and RNNs, as it requires access to only the last hidden state (feature vector) and the embedding layer from the network. A further restriction is that the feature vector used in the output layer must have the same dimension as the embedding feature vector. For the Transformer, to handle the dimensionality issue, we average the Transformer's representations before the output layer as the feature vector.

4 Experiments

4.1 Datasets

The summary of the datasets is shown in Table 1.

IMDB reviews were proposed by (Maas et al., 2011) for sentiment classification from movie reviews. It consists of two classes, i.e., positive and negative sentiments.

AGnews was proposed by (Zhang et al., 2015a) for researchers to test machine learning models for news classification. It consists of four classes (sports, world, business, and sci/tech).

DBpedia ontology classification dataset proposed by (Zhang et al., 2015b) consists of 15 non-overlapping ontology classes [1].

HealthLink constructed by Alberta Health Services, Canada. It contains a set of text transcripts written by registered nurses while talking with callers to the Tele-Health service in real-time. It consists of 2 classes ("go to hospital" and "home care"), and each class can be sub-categorized into sub-classes. This dataset will be available based on request.

Data set	Classes	Max length	Train size	Test size	Vocabulary size
IMDB	2	50	25000	25000	10000
HealthLink	2	20	60475	15119	23174
DBpedia	15	32	5600	63000	50002
AGnews	4	20	102080	25520	59706

Table 1: Summary of the datasets used in our experiments

4.2 Baselines

We compare our approach with several models for text classification including Transformers (Vaswani et al., 2017), IndRNN (Li et al., 2018), BLSTM (Zhou et al., 2016), hierarchical attention (Yang et al., 2016), LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014).

Transformer employs a multi-head self-attention mechanism based on scaled dot-product attention. We use only the encoder layer, and average the new representations before arriving in the classification's output layer.

IndRNN is an improvement over RNNs, where neurons in the same layer are independent of each other and connected across layers. We use the last hidden state as the feature vector.

Bi-LSTM employs an attention-based bidirectional mechanism on the LSTM network, which captures the salient semantic information (word-attention) in a sentence. These attentions enable the network to attend differently to more and less critical content when learning the representation. The last hidden state is used for classification.

Hierarchical attention provides two levels of attention mechanisms applied to the word and sentence level. In this paper, we use a sentence level-attention mechanism applied on a Bi-LSTM. The feature vector for classification is based on aggregating the hidden representation values (following the authors' implementation).

LSTM and GRU process the input word by word, and the last hidden state is used as the feature vector for classification.

[1]Because of computation time, we experimented with only a small number of samples.

4.3 Network configuration and training

We tokenize sentences and use the top N words that appeared in every instance for the vocabulary size. We did not use any pre-trained embeddings, and thus we randomly initialized the embedding layer. We also randomly initialized the **CSV**s. We did not use any hyper-parameter tuning on the validation as we are not focusing on achieving state-of-the-art predictive accuracy. Instead, we want to show that our method can achieve similar/better performance to the black-box, and provides a better explanation than existing approaches. The word embedding, semantic vector, and feature vector (at the output layer) dimensions are 128. For training each network, we use the Adam optimizer (Kingma and Ba, 2017) with a batch size of 64 and a learning rate of 0.0001. We also used a dropout with a probability 0.5.

4.4 Results and analysis

4.4.1 Classifier performance

We trained six different models (architectures) on four datasets. We have tried different values as the weight of each proposed loss term. The results depicted in Table 2 show that our semantic distillation approach captures more useful information from the training data than the baseline black-box. Our smaller model outperforms the black-boxes on all datasets, achieving better performance than the black-box. The new optimization problem does not affect the performance of the black-box model (see BBO (Black-Box with our new Objective function) in Table 2).

4.4.2 Explainability

In this part of our experiments, we focus on local explanations for text classification, i.e., explaining the output made by our proposed nearest neighbor classifier using **CSV** for an individual instance. Local explanations should exhibit high local fidelity, i.e., they should match the underlying model behavior. We evaluate our technique against the following methods:

	IMDB				AGnews				Dpedia				HealthLink			
Transformer (Vaswani et al., 2017)																
	F1	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**
Black-box	0.7703	0.7703	0.7703	0.7703	0.8794	0.8798	0.8796	0.8796	0.8653	0.8655	0.8655	0.927	0.6642	0.6645	0.6641	0.6647
BBO	0.7785	0.7787	0.7786	0.7786	0.8831	0.8836	0.8835	0.8835	0.8799	0.8802	0.8799	0.943	0.6887	0.6896	0.6887	0.6894
DISK-CSV	**0.8117**	**0.8187**	**0.8187**	**0.8187**	**0.9038**	**0.9039**	**0.9041**	**0.9041**	**0.8806**	**0.8811**	**0.8809**	**0.9438**	**0.7216**	**0.722**	**0.722**	**0.7216**
Attention-based Bi-LSTM (Zhou et al., 2016)																
Black-box	0.7961	0.7993	0.7966	0.7966	0.8887	0.8888	0.887	0.8888	0.843	0.8443	0.8434	0.9037	0.6706	0.6706	0.6705	0.6708
BBO	0.798	0.7999	0.7983	0.7983	0.8929	0.8941	0.8928	0.8927	0.8772	0.8774	0.8772	0.9399	0.6704	0.6705	0.6706	0.6705
DISK-CSV	**0.8025**	**0.8025**	**0.8025**	**0.8025**	**0.8956**	**0.8955**	**0.896**	**0.896**	**0.8812**	**0.8816**	**0.8815**	**0.9445**	**0.7207**	**0.7207**	**0.7209**	**0.7208**
IndRNN (Li et al., 2018)																
Black-box	0.776	0.7761	0.776	0.776	0.8773	0.878	0.8769	0.8769	0.8763	0.8765	0.8765	0.9391	0.6808	0.6814	0.6813	0.6808
BBO	0.7805	0.7858	0.7814	0.7814	0.8845	0.8847	0.8847	0.8848	0.8845	0.8889	0.888	0.9515	0.6808	0.6814	0.686	0.6869
DISK-CSV	**0.8018**	**0.8022**	**0.802**	**0.802**	**0.9025**	**0.9026**	**0.9028**	**0.9028**	**0.8887**	**0.889**	**0.8889**	**0.9524**	**0.7162**	**0.7184**	**0.7174**	**0.7164**
Hierarchical recurrent net (Yang et al., 2016)																
Black-box	0.7917	0.7919	0.7917	0.7917	0.8845	0.8855	0.8846	0.8846	0.847	0.8475	0.8467	0.9073	0.6708	0.6708	0.609	0.671
BBO	0.7808	0.7844	0.7813	0.7814	0.8874	0.8876	0.8874	0.8876	0.8709	0.871	0.8709	0.933	0.6829	0.6833	0.6833	0.6829
DISK-CSV	**0.8146**	**0.8146**	**0.8146**	**0.8146**	**0.9013**	**0.9013**	**0.9016**	**0.9016**	**0.8794**	**0.8796**	**0.8797**	**0.9425**	**0.7156**	**0.7158**	**0.7159**	**0.7157**
LSTM (Hochreiter and Schmidhuber, 1997)																
Black-box	0.745	0.7456	0.7452	0.7452	0.8711	0.8712	0.8714	0.8715	0.6597	0.7187	0.6445	0.6905	0.5922	0.6155	0.6044	0.6006
BBO	0.7461	0.7488	0.7466	0.7466	0.8745	0.875	0.8745	0.875	0.7993	0.8129	0.797	0.8593	0.6127	0.6718	0.6717	0.6712
DISK-CSV	**0.7912**	**0.7913**	**0.7912**	**0.7912**	**0.9005**	**0.9005**	**0.9009**	**0.9009**	**0.8657**	**0.8667**	**0.8662**	**0.928**	**0.7171**	**0.7178**	**0.7177**	**0.7171**
GRU (Cho et al., 2014)																
Black-box	0.748	0.7493	0.7483	0.7483	0.8709	0.8708	0.8711	0.8712	0.6537	0.7006	0.6442	0.6902	0.6106	0.6266	0.6187	0.6165
BBO	0.74483	0.753	0.7493	0.75	0.8847	0.885	0.8851	0.8906	0.8193	0.8123	0.812	0.875	0.6478	0.6572	0.6522	0.6562
DISK-CSV	**0.8069**	**0.8069**	**0.8069**	**0.8069**	**0.9046**	**0.9047**	**0.9049**	**0.9049**	**0.8831**	**0.8834**	**0.8833**	**0.9041**	**0.7154**	**0.7159**	**0.7158**	**0.7154**

Table 2: Comparison of our test performances with the baseline neural architectures on four datasets. Our nearest neighbour classifier achieves better performance than the black-box models. For the black-box models, we followed the implementation proposed by the authors of each baseline.

- **Random**. A random selection of words from the input sentence.
- **LIME** (Ribeiro et al., 2016) is a model-agnostic approach which involves training an interpretable model such as a linear model on instances created around the specific data point by perturbing the data. We evaluated by training the linear classifier using ~ 5000 samples.

We show the effectiveness of our method in explaining the prediction on three architectures (Transformer, IndRNN and hierarchical attention network) in Figures 3-8.

Automatic evaluation. We use model-agnostic evaluation metrics to demonstrate the effectiveness of our approach. (Nguyen, 2018) found that human evaluation correlates moderately with automatic evaluation metrics for local explanations. Hence, in our experiments, we use the idea of automatic evaluation to verify whether or not our explanations are faithful to what the model computes. We measure the local fidelity by deleting words in the order of their estimated importance for the prediction, then evaluate the change in F1 score w.r.t. the predicted class when no word is deleted. This type of evaluation is similar to other metrics used for model interpretation (Nguyen, 2018; Arras et al., 2017) except that we use F1 instead of classification accuracy. Results are shown in Figures 3-4. We obtained the plots by measuring the effect of word deletions and reporting the F1 when the classifier prediction changes. A larger drop in F1 indicates that the method could identify the words contributing most towards the predicted class by our classifier. Through Figures 3, 4 and 5, we can clearly see that our approach is capable of identifying the most salient features better than LIME. Please note that LIME requires probabilities (as the classifier's output), and hence we convert the outputs made by our nearest neighbor into valid probabilities.

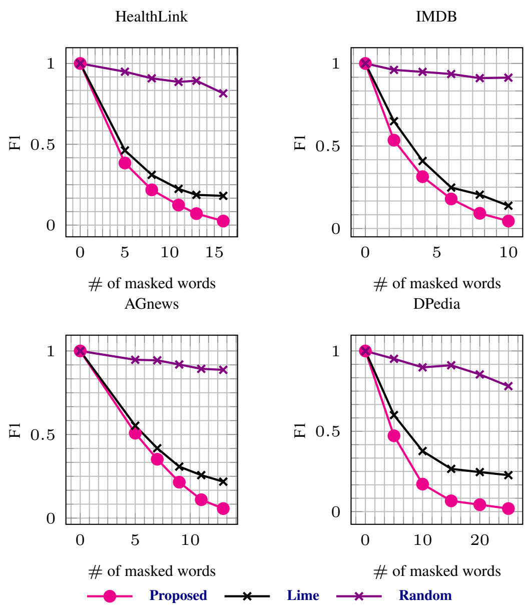


Figure 3: Change of F1 according to the number of masked important words. (Teacher model: Transformer)

Change in log-odds. Another automatic metric for evaluating explainability methods is to observe the change in the log-odds ratio (for the output probabilities). This metric has been used for a model's explanations (Shrikumar et al., 2017b; Chen et al.,

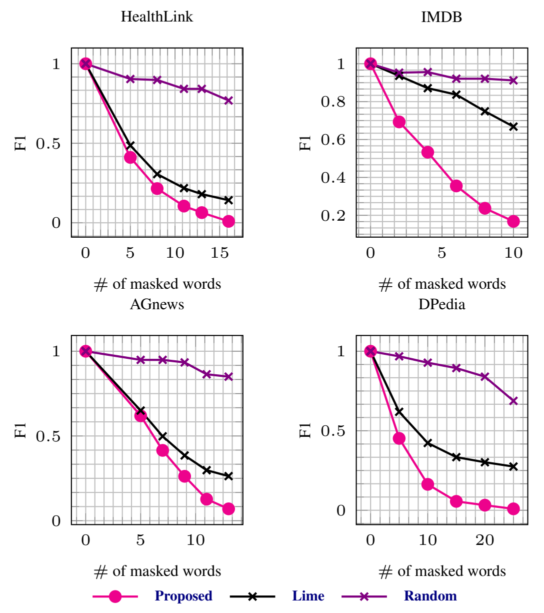


Figure 4: Change of F1 according to the number of masked words. (Teacher model: INDRNN)

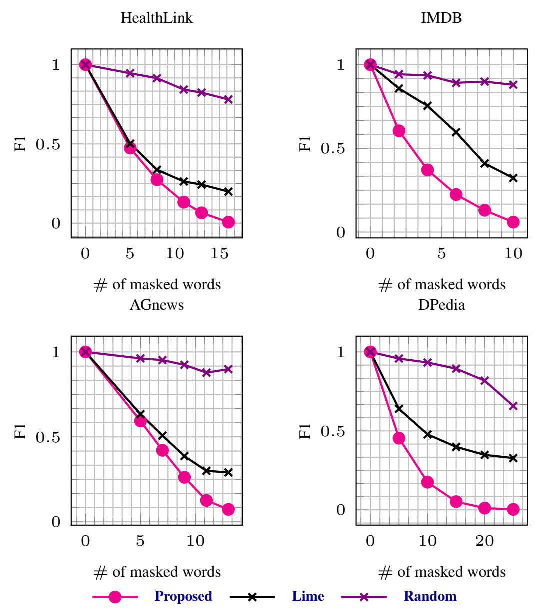


Figure 5: Change of F1 according to the number of masked words. (Teacher model: Hierarchical attention network)

2018). We normalize the cosine distance into valid probability distributions. This metric requires no knowledge of the underlying feature representation, and it requires access to only the instances. A log-odds ratio is a fine-grained approach, as it uses actual probability values instead of the predicted label as used in the previous experiment. But like the previous experiment, instead of tracking the change in F1, we observe the change in probabilities. We mask the top k features ranked by semantic similarity, and zero paddings replace those masked words. We then feed the input and measure the drop of the value between the target class's probability when no word is deleted and when k words are removed. Results are shown in Figures 6-8 reveal the effectiveness of our approach in capturing the words that affect the classifier's prediction. The experimental results show that our method delivers more insightful explanations than LIME.

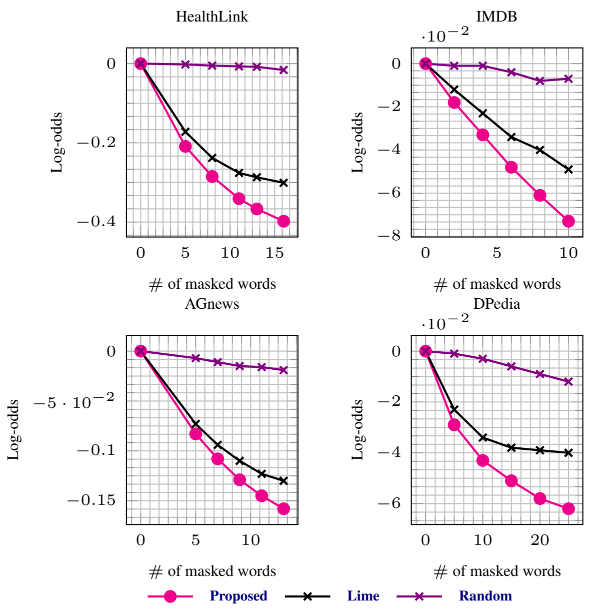


Figure 6: Change of log-odds according to the number of masked words. Lower log-odds scores are better. (Teacher model:Transformer)

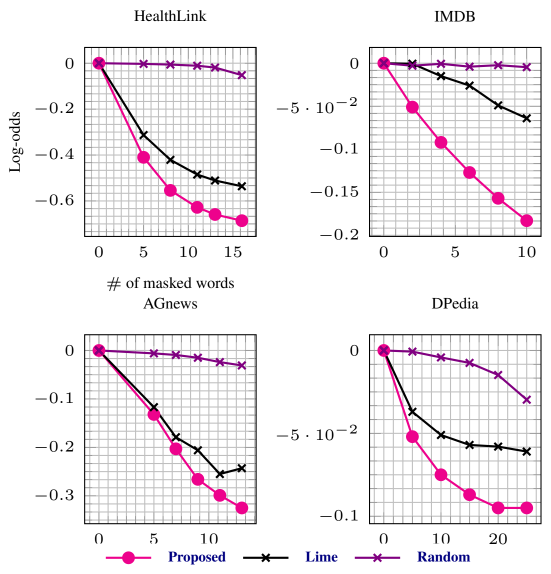


Figure 7: Change of log-odds according to the number of masked words. (Teacher model: INDRNN)

Interactive explanations. In some cases, end-users are interested in understanding the contribution of phrases instead of words. In addition, an end-user might be interested in understanding the

Example	Pos	Neg	Example	Home care	Go to hospital	Example	Home care	Go to hospital
Good	0.756	-0.752	Cough	-0.984	0.984	Fever	0.933	-0.932
Not good	-0.151	0.144	Bad cough	-0.993	0.993	Bad fever	-0.962	0.952
Very good	0.877	-0.878	Cough+sore throat	-0.995	0.995	fever+headache	0.929	-0.929
Sucks	-0.607	0.607	Chest pain	-0.959	0.958	Cold	0.168	-0.170
Not sucks	0.688	-0.681	Mild chest pain	-0.861	0.861	Cold+chest pain	-0.934	0.934
Just sucks	-0.825	0.828	Chest pain+high blood pressure	-0.991	0.991	Cold+fever	-0.532	0.961
Sucks but very good	0.255	-0.262	Breathing	-0.968	0.968	Blood pressure	-0.980	0.980
Heart-warming	0.335	-0.3444	Breathing difficulty	-0.992	0.991	Bad blood pressure	-0.990	0.990
Heart-warming+entertaining	0.538	-0.54	vomiting+breathing	-0.883	0.883	High blood pressure	-0.981	0.981

Table 3: XAI capability. Explaining word/phrase contributions and also providing contributions to other classes (semi-counterfactual explanation).

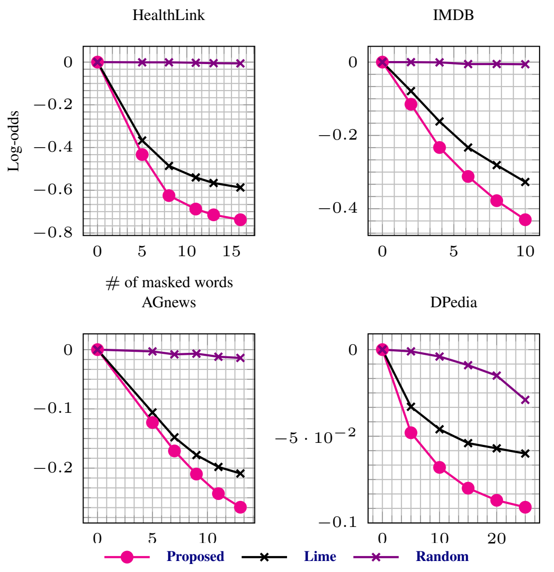
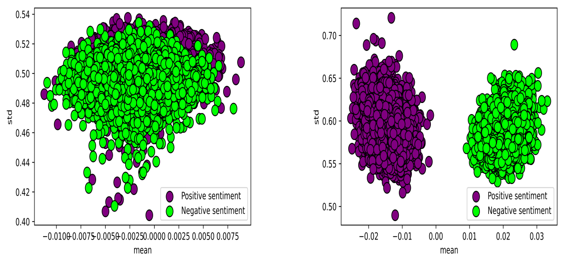


Figure 8: Change of log-odds according to the number of masked words. Lower log-odds scores are better. (Teacher model: Hierarchical attention network)

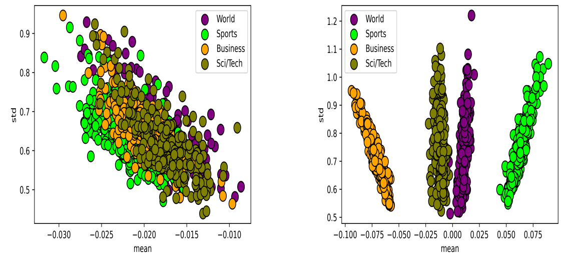
contribution of a word/phrase w.r.t. other classes, not only to the predicted class (semi-counterfactual explanation). Our model's results in support of these interests are shown in Table 3. Our technique can identify the contributions of phrases instead of words, and it can provide evidence w.r.t. other classes. For example, our method can recognize that "bad cough" has a stronger semantic contribution than "cough" w.r.t. the label "going to the hospital." Our approach can also recognize the difference between "mild chest pain" and "chest pain." In sentiment analysis, our method understands that "good" contributes to a positive sentiment while "not good" contribute to negative sentiment. Also, note that "very good" contributes more importantly to a positive sentiment than "good."

Clustering textual data. Another feature of the **CSV** classifier is the ability to cluster documents via **CSV**s without using dimensionality reduction techniques such as PCA clustering algorithms. Results of document clustering based on the distilled knowledge from the Transformer on the IMDB and AGnews are shown in Figure 9 and 10. The clusters explain our classifier's behavior and hence provide a global explanation of the model's prediction. We also show the critical role of using the pairwise Euclidean distance in our classification by clustering sentences into their predicted classes.



(a) Without Euclidean distance loss. (b) With Euclidean distance loss.

Figure 9: Sentence clustering on the predicted class using pairwise distance vs. without pairwise distance. (Dataset:IMDB)



(a) Without Euclidean distance loss. (b) With Euclidean distance loss.

Figure 10: Sentence clustering on the predicted class using Euclidean distance vs. without Euclidean distance. (Dataset: AGnews)

Parameter reduction. We compare the number of parameters used by our nearest neighbor classifier and that of the black-box approach, using the HealthLink data in Table 4. The number of parameters used by our compressed classifier is less than that of each black-box. Our model relies only on the embeddings and the **CSV**s, and the rest of the layers are dropped. The number of parameters of the proposed classifier is the same for all architectures because our classifier has the same size of the embedding layer and **CSV**s on each black-box architecture. Our model also reduced the inference time from $0.037 - 0.085$ seconds to 0.007 seconds, as shown in Table 4.

Method	# of parameters	# of dropped parameters	Inference time
Transformer			
Black-box	3049602	83074	0.085
DISK-CSV	**2966528**		**0.007**
IndRNN			
Black-box	2991490	24962	0.037
DISK-CSV	**2966528**		**0.007**
Attention-based bi-LSTM			
Black-box	3229826	263298	0.056
DISK-CSV	**2966528**		**0.007**
Hierarchical recurrent network			
Black-box	3114754	148226	0.039
DISK-CSV	**2966528**		**0.007**

Table 4: Number of parameters used for black-box and our proposed model. We also compare the inference time.

Semantics. We compare our proposed method's performance with and without capturing the semantic information (Equation 2). Results depicted in Table 5 show the importance of encoding semantic into the class discriminative vector.

	Proposed				**Without semantic**			
	F1	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**
Dpedia	0.8806	0.9438	0.8811	0.8809	0.0425	0.624	0.0693	0.0582

Table 5: The impact of first term in Equation 2 on the classifier's performance

Analyzing words. We are interested in what kind of words contribute most to the class prediction. For this analysis, we exploit the word-level sentiment annotation (Opinion Lexicon) provided by Liu [2] to track the top 10 words whose importance was the highest when predicting the sentiment class in the IMDB dataset. We evaluated the number of words contributing to each of the negative and positive sentiments on 1000 movie reviews. Table 6 shows that our approach can identify more salient words that lead to correct sentiment classification, i.e., our method can pick better sentiment lexicons than LIME.

	Proposed	**Lime**	**Random**
Positive sentiment	**597**	423	286
Negative sentiment	**382**	353	236

Table 6: The number of words in each sentiment class for 1000 samples from the test set.

4.5 Discussion

We have shown that semantic information can be extracted and used to create a simple interpretable/explainable classifier that performs better than the target black-box models. This simple classifier has the following proprieties:

- It captures the discriminative representations encoded by the black-box and encodes them in the **CSV**.
- For text classification, the distance is the lowest between the text input and the **CSV** of the correct class, and is high for the other **CSV**s of the incorrect classes.

[2] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

5 Conclusion and Future Work

We have explored an approach to knowledge distillation concurrently from any black-box model to produce a simple, explainable classifier. The distilled model achieves better results than the original black-box models in terms of the model's performance. Also, we showed that our distilled model provides a better explanation than LIME.

We have also proposed new types of explanations: First, a user can query with any-length phrases and receive feedback about the phrase's contribution to the classes. Second, we also provide word(feature) importance to non-predicted classes, which can be used as a semi-counterfactual explanation. Third, we showed how we could cluster the documents without employing the existing clustering method.

In future work, we would like to extend this idea to pre-trained networks, and we also plan to more deeply investigate the value of counterfactual explanations.

Acknowledgements

We acknowledge support from the Alberta Machine Intelligence Institute (AMII), from the Computing Science Department of the University of Alberta, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis in ACL*, pages 159–168.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, 10(7):e0130140.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of ACL*, pages 2963–2977.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. 2018. L-shapley and c-shapley: Efficient model interpretation for structured data. *ICLR 2019*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy L. Ba. 2017. Adam: A method for stochastic optimization. cornell university library. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*, pages 107–117.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5457–5466.

Wang X. Liu, X. and S. Matwin. 2018. Improving the interpretability of deep neural networks with knowledge distillation. In *Proceedings of IEEE International Conference on Data Mining Workshops*, pages 905–912.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, pages 142–150. Association for Computational Linguistics.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of NAACL-HLT, Volume 1 (Long Papers)*, pages 1069–1078.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Cynthia Rudin. 2018. Please stop explaining black box models for high stakes decisions. *32nd Conference on Neural Information Processing Systems (NIPS 2018), Workshop on Critiquing and Correcting Trends in Machine Learning.*

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017a. Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning*, pages 3145–3153.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017b. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of International Conference on Machine Learning (ICML)*, page 3319–3328.

Raphael Tang, Ashutosh Adhikari, and Jimmy Lin. 2018. Flops as a direct optimization objective for learning sparse neural networks. *arXiv preprint arXiv:1811.03060*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. 2018. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.