SoftCompose: Handling Compliance Violations in Advertisements with Neuro-symbolic Classifiers

Anonymous Author(s)

ABSTRACT

Online advertisement systems need to ensure that displayed advertisements provide a safe experience to end users compliant with legal regulations. Regulatory compliance is typically done using ML classifiers, but the drift between the training distribution and deployment distribution leads to undesirable deterioration in compliance during deployment. At deployment time, an advertisement from one of the regulated classes such as Adult, Tobacco or Firearm may get misclassified and consequently declared safe to display. In such a case, pragmatics force administrators of the advertisement system to write keyword-based rules to override the ML classifier's output for this and similar instances. In current industrial practice, such rules are composed with the ML model with a "hard composition" operator, where the rules always override the ML model when applicable. However, it is difficult to write the override rules accurately, and administrators often tend to write rules that are erroneously too specific to the current instance (and miss other related instances) or too general (and hence incorrectly cover and override the results for unrelated instances). As a result, the hard composition of the ML model with rules results in poor performance.

In this paper, we propose a novel "soft composition" operator to compose the ML model and the rules, where the composition operator algorithmically decides the set of points where each override rule fires. Our soft composition operator uses ground truth labels from the training set to automatically restrict or generalize the set of points where each override rule should be applied, so as to maximize performance. We also show how to efficiently calculate such a soft composition operator as more rules get added online. We also propose a compliance metric to evaluate the effectiveness of any such composition operator. We show that our proposed soft composition operator scores well on this compliance metric as well as overall accuracy, when compared to the hard composition operator, which is the current industrial practice.

CCS CONCEPTS

• **Information systems → Content match advertising**; • **Computing methodologies → Online learning settings**.

KEYWORDS

neuro-symbolic, distributional shift, rule modelling, compliant advertisements, online adaptation

ACM Reference Format:
Anonymous Author(s). 2022. SoftCompose: Handling Compliance Violations in Advertisements with Neuro-symbolic Classifiers. In *Proceedings of WWW Conference (WWW'23).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Online advertisement systems use machine learned text classifiers for assigning categories that feed into critical decision-making processes such as whether to display a specific advertisement. Because of the advantage of pre-trained language models, supervised deep neural networks are now the default choice for such text classification. However, in online deployments, a model once trained could drift from the training distribution due to the evolving nature of advertised products and their associated creative trendy text. This results in deterioration in accuracy of the deployed classifier, and more seriously, in the violation of legal regulations.

For example, we may get an advertisement that is from a regulated class (such as Adult, Tobacco or Weapon) that is misclassified as safe by the ML classifier and is hence displayed in violation of legal regulation. Conversely, we may get an advertisement that is perfectly acceptable from a regulatory perspective, but gets blocked because it is misclassified as a regulated class, resulting in lost advertisement revenue. In both cases, pragmatics force administrators to write rules to override the ML classifier's output and correct it.

The override rules are often expressed as keywords: text with keyword "cannabis" should be categorised as "Drugs". However, it is hard to write such override rules accurately for natural text. Sometimes the rules turn out to be overly general and capture more instances than what was intended. For example, Table 3 shows a rule where the occurrence of the word "tobacco" is used to classify a matching document with label "Tobacco". This rule is overly general and ends up misclassifying the text "...Shop now at tobaccomotorwear.com. Tobacco uses..", which is about a motorcyle clothing store. Conversely, some symbolic rules lack robustness by being specific, and miss capturing intended instances. For example, the phrasal rule "black jack" is too specific and misses classifying the text "..casino Miami Lakes.More Gaming.. ", even though the words "casino" and "gaming" are semantically related to "black jack". Table 4 shows more examples of such rules.

Current industrial practice is to compose the ML model M with an ordered set of override rules R, that are admittedly noisy due to difficulty in writing accurate rules, in a "hard" manner. We use $M \oplus_H R$ to denote such a "hard composition", where the rules in R always override the ML model M. Since rules are inherently non-robust and noisy, hard composition results in poor performance.

In this paper, we propose a novel "soft composition" operator $M \oplus_S R$, to algorithmically tune and produce a combined model over the decisions made by both M and R. Specifically, $M \oplus_S R$ carefully tunes the space where the rules R and ML model M get applied based on expected ground truth compliance labels in the training set, to minimize compliance errors. We expect the feedback to arrive in an online manner as the system gets deployed and drifts away from the training distribution. Therefore, the rules needs to be integrated to the composition $M \oplus_S R$ efficiently in an online manner. Such efficient online updates are key in real-world systems due to the large scale of ML model and rules, and the desire to push updates with low latency to keep the system up and running. While there exists extensive prior work on integrating logical rules with ML classifiers [3, 5, 9, 13, 17], most of these focus on training a neural model using rules as additional supervision. In contrast we refrain from online modification of parameters of M since it may lead to unintended, non-local changes and incur runtime overheads given the large scale of M and R. Instead we propose a novel method of softening hard rules and online learning to restrict or generalize them while incurring low latency. Also, our method of softening preserves the interpretability of rules.

We also propose a compliance metric to evaluate the effectiveness of any such composition operator that is centered around override rules. Standard error metrics treat all examples identically, while in a operational system, predictions on examples already correctly covered by rules or ML classifiers, should not be worsened for a combination strategy to be considered reliable by an administrator. Our compliance metric is designed to target such a requirement. We perform experiments on four industrial datasets on serving advertisements. In addition, for reproducibility, we also report experiments on three public non-ad text datasets spanning question classification, Q&A, and ontology creation tasks. We compare our proposed soft composition operator with the hard composition operator (which is current industrial practice), as well as one other baseline (called KNNMT) and show that our soft composition operator scores well on this compliance metric as well as overall accuracy.

Contributions. Overall our main contributions are: (1) Motivated by the practical challenge of regulatory compliance in online advertisement systems, we formulate the problem of online incorporation of noisy feedback rules to a trained ML text classifier in response to changing data distribution. (2) In contrast to the current practice of applying feedback rules as hard overrides, we propose a new online algorithm that denoises hard rules by softening and shaping their match by learning two parameters per rule. (3) We propose a new compliance metric for evaluating the efficacy of composition operators in rule adherence while accounting for their noise. (4) We evaluate our proposed method and other baseline using industrial online advertisement data sets, as well as three public data sets.

2 PROBLEM FORMULATION

Let $\mathcal{X}$ denote the space of instances and $\mathcal{Y} = \{1, \ldots, K\}$ denote the space of class labels. We are given a set of labeled examples $D = \{(\mathbf{x}_1, \ell_1), \ldots, (\mathbf{x}_n, \ell_n)\}$. A neural classifier $M : \mathcal{X} \mapsto \mathcal{Y}$ trained on this dataset is available for deployment. During deployment time, the administrator may decide to override an arbitrary set of misclassified instances. Along with providing the correct label of the instance, the administrator generalizes the supervision by providing a symbolic rule. We denote the accumulated feedback set at time t as $R_t = \{(r_1, \ell_1, \mathbf{e}_1), \ldots, (r_t, \ell_t, \mathbf{e}_t)\}$ where r_j is a symbolic rule that seeks to assign $\ell_j \in \mathcal{Y}$ to examples on which it matches, and $\mathbf{e}_j \in \mathcal{X}$ is the instance which triggered the override and is referred as the exemplar of the rule r_j. For the task of text classification, we consider phrasal rules with non-contiguous n-gram words that fire if the words in the rule are present in the text. Examples of such rules appear in Table 3 and Table 4.

The rules provided as feedback need to be instantly integrated into future predictions. Instead of modifying the parameters of M via fine-tuning, which may lead to non-local unexpected changes and incur runtime overheads, we seek to design a combined model $C_t = M \oplus R_t$ that implements on-the-fly ensembling of M and the feedback rules in R_t.

Current industrial practice is to design a a "hard" composition operator as follows: for a test instance $\mathbf{x}$, find the set of rules which fire on this instance using an exact phrasal match and perform a consensus among them to predict the output label. If no rule fires, then use M to perform the classification. We denote such a baseline composition operator as HardCompose or $\oplus_H$. The main limitation of such a composition is that the override rules are noisy, in general. In some cases, the rules are too general and inadvertently cover unrelated instances (see Table 3). In other cases, they are too specific and miss covering related instances (see Table 4). Next, we present a new operator to compose rules and ML models better.

3 OUR APPROACH: SOFT COMPOSITION

We design a soft composition operator SoftCompose or $\oplus_S$, which has better compliance and accuracy properties than the baseline composition operator. Our proposal has two components: (1) A novel rule matching function that makes specific rules more robust by generalizing exact keyword match to similarity in the embedding space, and restricts general rules by exploiting the context of the keyword in the exemplar, and (2) An online algorithm that learns to control the firing of each rule and achieves consensus with the ML model by learning two parameters per rule using labeled instances.

3.1 Soft Matching Instances to Rules

We need to design a matching score that can cater to both rules that are too specific and rules that are too general. A simple method would have been to replace hard matches with similarity in the embedding space that preserves semantic similarity. However, this method would not help with restricting rules that are too general. For these cases we depend on the exemplar $\mathbf{e}_j$ attached with each rule r_j to capture the intended context in which the words in the rules were meant to fire. This led us to design a new hybrid embedding of each w in an example $\mathbf{x}$ that concatenates embedding of a word w by concatenating the word's normalized non-contextual embedding $E_{nc}(w)$ with the normalized contextual embedding $E_c(w, \mathbf{x})$ in $\mathbf{x}$.

$$E(w, \mathbf{x}) = [E_{nc}(w), E_c(w, \mathbf{x})]$$

The non-contextual embedding $E_{nc}(w)$ of the word is obtained from the 300-dimensional Glove embeddings[1]. The contextual embedding $E_c(w, \mathbf{x})$ is obtained by passing the example $\mathbf{x}$ through the fine-tuned model M and taking the word's contextualized embedding. If the word is tokenized in smaller sub-tokens, we take the average of the embeddings of the sub-tokens.

With the above design, the contextual embeddings help to capture a broader context in which the phrase of the rule appeared in its exemplar. However, contextual embeddings alone do not suffice since we have only one exemplar per rule and irrelevant context may corrupt similarity for words where more precise match was intended. For example, unigram rules like "where" for the "Location" class in a Question classification task may fire on questions containing "why" or "what" that have a similar context if we only use contextual embeddings.

The similarity between a rule word w in context $\mathbf{e}$ and a word x appearing in a test instance $\mathbf{x}$ is computed as the average of the cosine similarities of their contextual and non-contextual embeddings, normalized to $[0, 1]$ range.

$$s(w, \mathbf{e}, x, \mathbf{x}) = \max(0, E(w, \mathbf{e})^\top E(x, \mathbf{x}))/2$$

Now for computing the soft match between a rule r_j comprising of multiple words $r_j^1, \ldots, r_j^n$ and an incoming text instance $\mathbf{x}$ with words $x_1, \ldots, x_m$, we use the following distance measure:

$$d(r_j, \mathbf{x}) = 1 - \min_{1 \le i \le n} \max_{1 \le k \le m} s(r_j^i, \mathbf{e}_j, x_k, \mathbf{x}) \quad (1)$$

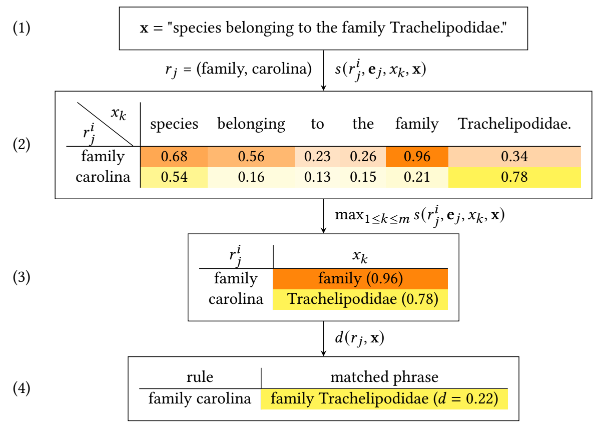
Each word in the rule is matched with a word from the test instance with the highest similarity. The similarity of the rule with the text is then taken as the minimum similarity over the matched pair of words which defined an "and" semantics over the words in the rule. So if some word in the rule does not have any similar word in the text, the overall similarity of the rule and the text becomes low. Figure 1 shows an illustration of the distance computation by our soft matcher.

3.2 Online Learning of Rule Parameters

With the distance metric defined, we learn two parameters for each rule in the system that control when and how the rules fire on an instance and how they combine with the ML model. We want the rule parameters to be determined efficiently in an online manner as rules arrive. Further, while we have access to the original labeled data D used for training the ML model, the deployment distribution may have shifted, so we cannot rely on labeled data to learn complicated non-linear transformations of the rule scores.

Based on the above considerations, we chose a simple two parameter model where for each rule r_j, we define two parameters $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_j$. The parameter $\boldsymbol{\alpha}_j$ controls the coverage of the rule. If $d(r_j, \mathbf{x}) < \boldsymbol{\alpha}_j$, the rule fires on $\mathbf{x}$. The second parameter $\boldsymbol{\beta}_j$ is used to convert the distance into a distribution over class labels so that the rule's output can be combined with the label distribution assigned by the ML model. We assume that the rule has an exponentially decaying probability distribution in the region of space

[1]If the word is not present in the vocabulary of Glove, then $E_{nc}(w) = \mathbf{0}$.


(1)
x = "species belonging to the family Trachelipodidae."
r_j = (family, carolina)
$s(r_j^i, \mathbf{e}_j, x_k, \mathbf{x})$
(2)
$\max_{1 \le k \le m} s(r_j^i, \mathbf{e}_j, x_k, \mathbf{x})$
(3)
$d(r_j, \mathbf{x})$
(4)

r_j^i \ x_k	species	belonging	to	the	family	Trachelipodidae.
family	0.68	0.56	0.23	0.26	0.96	0.34
carolina	0.54	0.16	0.13	0.15	0.21	0.78

r_j^i	x_k
family	family (0.96)
carolina	Trachelipodidae (0.78)

rule	matched phrase
family carolina	family Trachelipodidae (d = 0.22)

Figure 1: Distance function illustration. (1) Compute similarity (s) between each pair of words between r_j^i and x_k. (2) For a given rule word, select the highest similarity among instance words. (3) Among the rule words, select the lowest similarity score. Compute distance as 1-similarity score. (The exemplar $\mathbf{e}_j$ is not shown to reduce clutter.)

covered by the rule. The distribution is parameterized with $\boldsymbol{\beta}_j$ as:

$$P_j(y \mid \mathbf{x}, \beta_j) = \begin{cases} e^{-d(r_j, \mathbf{x})/\beta_j}, & \text{if } y = \ell_j \\ \frac{1 - e^{-d(r_j, \mathbf{x})/\beta_j}}{K-1}, & \text{otherwise} \end{cases} \quad (2)$$

The above formula assigns a probability of one for the rule's label ℓ_j when an example is very close to r_j, and for examples far from r_j the probability is uniformly distributed across all labels. We combine the probability assigned by all the rules by weighting them with the softmax distribution on the basis of the coverage of the rules:

$$P_R(y \mid \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_j \frac{\exp(-(d(r_j, \mathbf{x}) - \alpha_j))}{\sum_k \exp(-(d(r_k, \mathbf{x}) - \alpha_k))} \cdot P_j(y \mid \mathbf{x}, \beta_j) \quad (3)$$

A weighted convex combination of the combined probability distribution of the rules is taken with the ML model's distribution to get the final probability distribution of the system over the classes. The weight assigned to the rules is determined using the highest coverage score:

$$P(R \mid \mathbf{x}, \boldsymbol{\alpha}) = \max_j \sigma(-(d(r_j, \mathbf{x}) - \alpha_j)) \quad (4)$$

$$P(y \mid \mathbf{x}, R, \boldsymbol{\alpha}, \boldsymbol{\beta}) = P(R \mid \mathbf{x}, \boldsymbol{\alpha}) \cdot P_R(y \mid \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) + (1 - P(R \mid \mathbf{x}, \boldsymbol{\alpha})) \cdot P_M(y \mid \mathbf{x}) \quad (5)$$

3.3 Online Training Algorithm

We learn the parameters α_j, β_j for each rule in an online matter as they arrive. Algorithm 1 presents an outline. At time t, assume t rules have arrived and we have learned parameters $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_j$ for $j \in \{1, \ldots, t\}$. Our goal is to add a new rule r_{t+1} to the system by learning its parameters and updating previous rules' parameters if necessary. As the intent of the system is to make local changes to the classification boundary, we only consider the points that lie in the sphere of influence of the new rule for learning the parameters.

Algorithm 1 Online Training of SoftCompose

```
Input: Model $M \oplus_S R_t$, Rule $r_{t+1}$, Original labeled data $D$
Output: Model $M \oplus_S R_{t+1}$
1: $D_t$ = Closest Instances$(e_{t+1}, D) \cup \{\mathbf{e}_1, \ldots, \mathbf{e}_{t+1}\}$
2: $C_t = \{r_{t+1}\}$
3: for Rule $r_j$ in $R_t$ do
4:     if $d(r_j, e_{t+1}) < \alpha_j$ then
5:         Add $r_j$ in $C_t$
6:     end if
7: end for
8: $L = \frac{1}{|D_t|}\sum_{i=1}^{|D_t|} -\log(P(y = \ell_i \mid \mathbf{x}_i, R_{t+1}, \boldsymbol{\alpha}, \boldsymbol{\beta}))$
9: Optimize $\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j$ over L for all $r_j$ in $C_t$
10: $M \oplus_S R_{t+1} = M \oplus_S (R_t \cup \{r_{t+1}\})$
11: return $M \oplus_S R_{t+1}$
```

Hence, we first construct a set of instances D_t that is the union of the set of exemplars from R_{t+1} and a set of $2k$ closest neighbors of the exemplar e_{t+1} in the original labeled set D. The closest $2k$ instances are computed using distance in Eq 1 such that half the instances have the same label ℓ_{t+1} as the rule and other half have a different label to ensure that we also get negative examples for training. From R_t, we define a set of conflicting rules C_t as the set of rules that fire on the exemplar e_{t+1}, including the latest rule r_{t+1}. We jointly optimize the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for all the conflicting rules C_t by minimizing the loss L, the expected negative log likelihood of the probability of the true label over the set of labeled instances in D_t. The parameters of the new rule r_{t+1} are initialized with 0.1 whereas for the old rules, we initialize with their previous learned values. The parameters are optimized via gradient descent on the loss for a fixed number of epochs.

3.4 Inference

For a text instance $\mathbf{x}$, we first compute the combined probability $P_R(y \mid \mathbf{x}, R, \boldsymbol{\alpha}, \boldsymbol{\beta})$ by taking the softmax weighted summation over the rules r_j that have $\sigma(-(d(r_j, \mathbf{x}) - \alpha_j)) > 0.5$. We consider all such rules to be firing on the instance. Finally, we compute the probability $P(y \mid \mathbf{x}, R, \boldsymbol{\alpha}, \boldsymbol{\beta})$ as given in equation (5) and predict the label that gets the highest probability.

4 EXPERIMENTS

We evaluate our method of soft composition against hard composition and KNNMT on seven datasets. In addition to the standard accuracy metric, we propose a new evaluation metric that is specifically suited to our goal of measuring compliance of a working system in response to newly added rules.

All the experiments were performed on a machine with 16 GB GPU size and 110 GB RAM. The experiments were performed using the Python3 programming language with the language models implemented using PyTorch and Hugging Face's Transformers library [16].

4.1 Compliance metric

The main goal of the compliance metric is to ensure that examples that are correctly classified by the hard firing of a feedback rule, should be penalized for misclassification by any proposed soft composition. However, examples that were correctly classified by the ML model, and therefore not overridden, are also an implicit compliance rule and should be penalized for misclassification. We combine these two penalties into a compliance error as follows:

Let $\mathcal{R}_C$ be the set of classified instances on which some rule in the set R fires and results in a correct classification. Let $\mathcal{M}_C$ be the set of instances that are correctly classified by the ML model M.

Let P_c be the set of instances that are correctly classified by the predictor as a whole. Compliance error of the predictor wrt the hard-rules system is defined as the set $(\mathcal{R}_C - P_C)$. Compliance error of the predictor wrt the ML-based system is defined as the set $(\mathcal{M}_C - P_C)$. We combine these errors to get overall compliance error as:

$$\text{CErr}(\mathcal{R}_C, \mathcal{M}_C, P_C) = \frac{|\mathcal{R}_C - P_C| + |\mathcal{M}_C - P_C|}{|\mathcal{R}_C| + |\mathcal{M}_C|} \tag{6}$$

Unlike overall accuracy that treats errors on all instances the same, this metric increases the weight of examples that were correctly classified by rules.

Datasets	|Train|	|Val|	|Test|	#Class	#Rules
AdsAddiction	139425	15492	17213	4	256
AdsDating	299837	33316	33316	3	306
AdsHealth	19880	2209	2455	3	68
AdsAdultWeapon	244379	36535	40595	3	2341
Yahoo	882000	140000	60000	10	4900
TREC	150	500	500	6	68
DBpedia	245148	56000	70000	14	1551

Table 1: Statistics of datasets. Except TREC, rules are generated for other datasets using Algorithm 2.

4.2 Datasets

We evaluate our rule composition method across four datasets from a commercial online advertisement system and three public classification datasets namely, Yahoo, DBpedia from [18] and TREC [8].

The first four datasets pertain to an online advertisement system, wherein it is critical to identify and flag advertisements that are not compliant with regulatory policies.

- **AdsAddiction**: Consists of 4 classes Drugs, Tobacco, Gambling and Neither.
- **AdsDating**: Consists of 3 classes Dating, SexualEnhancements and Neither.
- **AdsHealth**: Consists of 3 classes Personal Hygiene, Supplements and Neither.
- **AdsAdultWeapon**: Consists of 3 classes Adult, Weapon and Neither.

The next three datasets are publicly released classification datasets.

- **Yahoo Answers!**: This is a topic classification dataset consisting of question content and best answer. The topics range across 10 different categories like Society & Culture, Science & Mathematics, Health, etc.
- **TREC**: This is a TREC-6 dataset to classify a question to one of six categories: Abbreviation, Entity, Description, Human, Location, Numeric-value. Following [3], we take 68 rules along with their exemplars.

- **DBPedia**: The DBpedia ontology dataset comprises of 14 non-overlapping classes from DBpedia 2014. The text to be classified includes the title and abstract of each Wikipedia article.

Table 1 summarizes the train, validation and test sizes across datasets along with the number of classes and rules. For the TREC dataset, we use the set of 68 rules generated by Awasthi et al. and for the other datasets. we simulate feedback rules using the procedure outlined in Algorithm 2. We plan to release the code and public datasets[2].

4.3 Shift data creation

To simulate distribution shift, we create an out-of-distribution shift set for each dataset as follows: For each class in the dataset, we cluster all of the training set using k-means with $k = 10$. For clustering, we use [CLS] embedding obtained after passing the datapoint through the pre-trained ML model. In other words, we perform clustering in the pre-trained embedding space of the model. We initialize the k-means algorithm using k-mean++ and set the maximum number of iterations to 300. We choose the best clusters (in terms of inertia) after running the k-means algorithm 10 different times with different initial centroid seeds. After this run, we get a set of 10 cluster centers for each class in the dataset.

We then partition these 10 clusters into two sets - the shift set and the unshift set. For creating the partition, we go through all possible partitions of the clusters and rank them according to the following metric:

$$\frac{||\mu_s - \mu_u||^2}{\sigma_s^2 + \sigma_u^2} \tag{7}$$

where: s is the shift set, u is the unshift set, μ_x is the mean of all vectors in the given set x, σ_x^2 is the mean of the variance of the vectors in the given set x along every dimension.

We pick the top 5 partitions of shift and unshift set and report results averaged on these 5 versions of the datasets. Note that the shift-unshift partition is only on the training set. The validation and test set follow the overall distribution which is a mix of shift and unshift distributions, and thus provides a form of test distribution shift. We train the ML model only on the unshift set.

Algorithm 2 Rule Generation Algorithm

```
Input: Model $M$, Shift set $D_{\text{shift}}$, precision lower bound $L$,
       precision upper bound $U$, support threshold $st$
Output: Final Rules Set $R$
1:  Let $R_t$ = empty set.
2:  for Instance $(\mathbf{x}_t, \ell_t)$ in $D_{\text{shift}}$ do
3:      If $(M \oplus_H R_t)(\mathbf{x}_t) == \ell_t$ , continue
4:      $C$ = Possible Phrasal Rules($\mathbf{x}_t$)
5:      $C$ = Filter Rules($C, U, L, st$)
6:      If $C$ is empty, continue
7:      $r$ = Sample a rule by weighing on precision($C$)
8:      Add $(r, \ell_t, \mathbf{x}_t)$ to $R_t$
9:  end for
10: return $R_t$
```

[2]Code is shared anonymously at http://aka.ms/AAicrqv for the reviewers

4.4 Rule Generation

If override rules are not available for a dataset, we generate the rules using Algorithm 2 on the shift set D_{shift}. We sequentially scan D_{shift} and obtain a rule on each instance $\mathbf{x}_t$ that is misclassified by $M \oplus_H R_{t-1}$ where R_{t-1} denotes the rules created before it. We obtain the set of all unigram and bigram phrases that can serve as potential rule for $\mathbf{x}_t$ while filtering the phrase (1) whose precision is outside a configurable range and (2) whose support is lesser than a constant st. Finally, we sample a rule by weighing based on the rule's precision. If no rules satisfy the constraints, then we do not add a rule for that instance. Lastly, after getting the set of rules for all the instances, we randomly subsample the set to keep the number of rules within a tractable range.

In our experiments, we keep the support threshold $st = 10$ and the precision range for rules as $[0.6, 0.9]$ and subsample the rules to keep the total number of rules within 10,000. We also perform ablation studies by varying the range of the precision in specific intervals ranging from low precision to high precision. The rule generation algorithm is described in Algorithm 2.

4.5 The ML Model

We use the pretrained uncased DistilBERT model [10] and fine-tune it on the unshifted training set. We use the Adam optimizer [7] to train the ML model. The learning rate, number of training steps and batch size used are given in the Appendix for all the datasets. The hyperparameter tuning procedure for SoftCompose and KNNMT is also given in the Appendix.

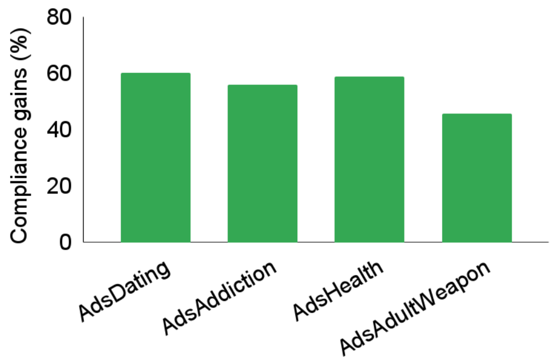


Figure 2: Compliance gains with SoftCompose **method over** HardCompose **on four advertisement data sets**

4.6 Comparing Soft and Hard Composition

In the HardCompose (denoted as $\oplus_H$) method, which is the current industry standard, if token(s) in the hard rule r (unigram or bigram) match exactly with the tokens in the instance, then the rule fires and the label associated with it overrides the ML model. Otherwise, the prediction is taken from the ML model. When multiple rules fire, the majority label is predicted. Since we designed the SoftCompose method as an alternative to the HardCompose method, we first compare these two approaches. We report numbers across both the advertisement data sets and public data sets using both these

Datasets	Compliance Error (↓)		Accuracy (↑)	
	HardCompose	SoftCompose	HardCompose	SoftCompose
AdsDating	2.95 (0.006)	**1.18**	94.07 (0.009)	**96.81**
AdsAddiction	8.64 (0.007)	**3.82**	82.04 (0.007)	**88.82**
AdsHealth	8.85 (0.012)	**3.66**	80.75 (0.017)	**87.47**
AdsAdultWeapon	10.49 (0.000)	**5.70**	78.98 (0.000)	**87.81**
Yahoo	5.62 (0.001)	**2.93**	68.64 (0.001)	**71.70**
TREC	22.45 (0.00)	**2.87**	55.76 (0.000)	**81.08**
DBpedia	8.58 (0.000)	**1.16**	85.10 (0.000)	**95.68**

Table 2: Comparison of SoftCompose **method with** HardCompose **on four advertisement data sets and three public datasets. We also report the paired-t test numbers with respect to** SoftCompose **method.** SoftCompose **improves upon** HardCompose **on all the datasets with respect to compliance and overall accuracy.**

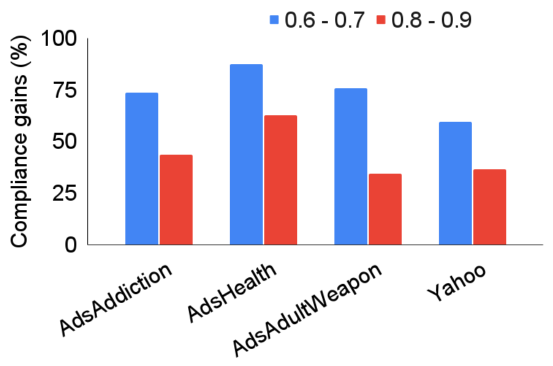


Figure 3: Comparison of compliance gains using SoftCompose **method with rules from different precision ranges, with respect to** HardCompose**. We observe that the compliance gains on rules with lower precision ranges is more than the gains on rules with higher precision ranges.**

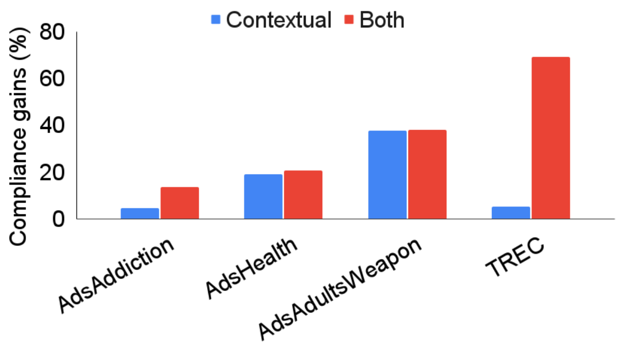


Figure 4: Compliance gains in the SoftCompose **method using contextual embeddings only and a combination of contextual and non-contextual embeddings, when compared to using non-contextual embeddings**

approaches. We report both the novel compliance metric (described earlier) as well as the overall accuracy.

Table 2 shows the comparison of the two methods. As we can see from the table, SoftCompose method results in the **lower** compliance error and **higher** accuracy on all the datasets. Figure 2 shows the percentage reduction in compliance error obtained by SoftCompose method over HardCompose method (using the data from Table 2). We denote this as *compliance gain*. As shown in the Figure, SoftCompose provides over 55% gains across the four advertisement datasets.

4.7 Ablation study

4.7.1 Varying rule precision ranges: In our evaluations presented in the Table 2, we used rules with a precision range of 0.6 – 0.9. In this ablation study, we ask how the performance of SoftCompose varies in presence of low precision rules, with precision of 0.6 – 0.7 and highly precise rules, with precision of 0.8 – 0.9

Figure 3 shows compliance gains on 4 datasets in different precision range settings, with respect to HardCompose. As seen from the figure, our experiments show that the compliance error with highly precise rules is smaller than the lower precision rules. This is as expected, as highly precise rules have reduced conflicts and do not cover unrelated instances. On the other hand, lower precision rules require further restriction in their α values. Another observation is that the reduction in compliance error with our SoftCompose method as compared to HardCompose method is significant with lower precision rules as compared to highly precise ones. For instance, in AdsAddiction and AdsHealth datasets, there is 73 – 87% reduction in compliance error with respect to HardCompose with rules in the lower precision range, as compared to 43 – 63% reduction with highly precise rules. We observe a similar trend in accuracy numbers. Table with accuracy numbers is included in the appendix . In the real world, administrators are more likely to introduce lower precision rules, as they often lack knowledge of past rules. And our proposed SoftCompose method results in higher compliance gains in such scenarios.

4.7.2 Using only contextual or only non-contextual embeddings: In our evaluations presented in the Table 2, we used a combination of contextual and non-contextual embeddings as the distance function. In this ablation study, we evaluate the effectiveness of our proposed SoftCompose method in presence of only one type

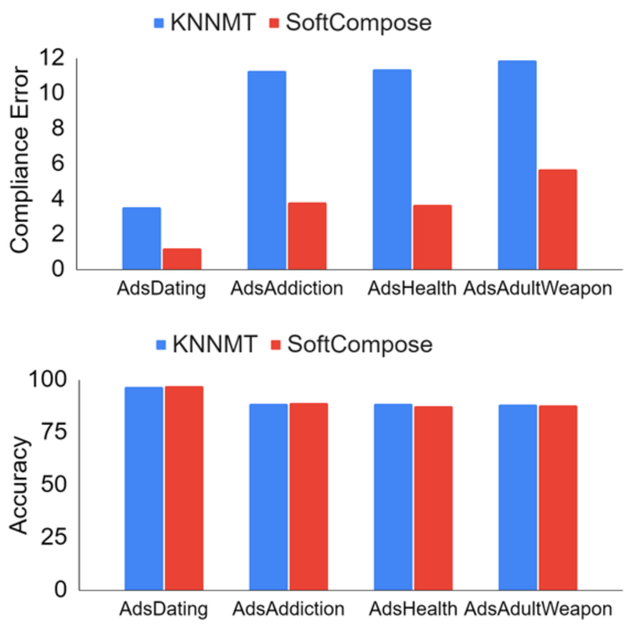


Figure 5: Comparison between KNNMT and SoftCompose.

of embedding. Figure 4 shows compliance gains over using only non-contextual embeddings in two settings: using only contextual embeddings and secondly, using a combination of contextual and non-contextual embeddings. Observe that the experiment with contextual embedding is more compliant as compared to the non-contextual run, demonstrated by positive gains.

For instance, for AdsAddiction dataset, the compliance gains with contextual embeddings over non-contextual embeddings is 4.63%. Using a combination of both type of embeddings results in compliance gain of 13.51% over non-contextual embeddings, showing the importance of combining both type of embeddings. This is as expected since non-contextual embeddings do not capture the broader context with respect to other words in the exemplars. At the same time, combining both types of embeddings fares better as it gets the best of both the worlds. Specifically, with non-contextual embeddings, we provide more importance to the tokens in the rule as compared to the context.

For a small dataset like TREC where there is not much scope of generalizing the rules, heavily depending on the contextual embedding may cause many rules to misfire. Therefore, using non-contextual embedding along with contextual provides a drastic compliance gain of 69% as can be observed in the Figure 4.

4.8 Comparing Soft Composition with KNNMT

The KNNMT method, an offline adaptation method, is an implementation of the algorithm proposed in [6] which treats the memory as a K-nearest neighbor classifier and the ML model as another classifier. The method is an offline method, and the adaptation is done solely using data points from feedback due to distribution shift and not rules. Hence, the method is not suitable for an "apples-to-apples" comparison with SoftCompose method. With this caveat, since KNNMT is an adaptation method, we compared the two approaches on the advertisement datasets, and the results are shown in Figure 5. As shown in Figure 5, the SoftCompose method results in significantly lower compliance errors on all the datasets. This is due to the KNNMT method using only the exemplar for adaptation and ignoring the feedback rule. As shown in Figure 5, the overall accuracy of the KNNMT method is comparable to the SoftCompose method, and in some cases marginally better. Our intuition is that this is due to the fact that KNNMT is an offline method, whereas we add the feedback rules incrementally, in an online manner for SoftCompose method.

The results show that the SoftCompose method, despite being an online algorithm, is as capable as the contemporary offline adaptation algorithms while being more compliant with the ML model and the feedback rules.

4.9 Anecdotal examples

Tables 3 and 4 show examples SoftCompose method improves over HardCompose and ML methods. Table 3 lists examples where a phrasal rule fires and assigns an incorrect label. For instance, phrasal rule "hand grenades" fires on the ads text "hand grenades : a handbook on rifle and hand grenades..." marking it as a Weapon, which is incorrect. The proposed SoftCompose restricts the rule from firing on this example, and assigns the correct label. Similarly, the unigram rule "tobacco" fires on "Best Armoured Jeans Restock Shop Now tobaccomotorwear.com. Tobacco uses....craftsmanship" marking it as "Tobocco", even though the ads text pertains to motorcycle clothing. Soft rule formulation restricts this rule from firing on this example and assigns it the correct class.

Table 4 lists examples where the phrasal rules are overly specific and do not fire on related examples. Also, ML model predictions on these examples are incorrect. The examples illustrate other challenges of multilinguality, spelling errors, *adversarial* injection of filler text, that fool exact match but succeed with soft match. As an example, in the example "cannibus seeds … Is The Newest Place to Search..", both phrasal rule and ML model fail to predict correctly, but SoftCompose method softens the token "vaping", and predicts "Tobacco" class correctly on this instance.

5 RELATED WORK

Detecting non-compliant or unsafe advertisement has been studied to some extent in the literature. Attenberg and Provost [2] propose a hybrid method of search and active labeling to build better models for detecting harmful ads. Dalvi et al. [4] propose building a model that is robust to any adversarial strategy, and Sculley et al. [12] propose to train a tiered model that incorporates both automated and semi-automated components, including leveraging human experts. Most of these works address the problem of model building. However, none of these prior works address the issue of distribution drift, and the idea of combining feedback rules and the machine learning model to address compliance violations at deployment that occur due to such a distribution shift. Our work is complementary to these efforts and can be utilized to improve these systems.

Our work can also be seen as being at the intersection of neuro-symbolic methods and online adaptation methods. While there is much related work in each of these topics separately (and we survey

Example	Rule phrase	Hard match label (incorrect)	SoftCompose label (correct)
Best Armoured Jeans Shop Now tobaccomotorwear.com. **Tobacco** uses..	tobacco	Tobacco	NotTobacco
... Vintage Corduroy 90s Shirt- **Tobacco** Coloured- Collared ...	tobacco	Tobacco	NotTobacco
Hewitt-Tville High School...has several varsity sports teams..**track and field**..	track field	Athlete	Education Inst.
Ursula ... author of novels .. short stories mainly in the genre of **science fiction**	science fiction	Written-work	Artist
The first USS Calypso was a steamer ...to prevent the South from **trading**...	trading	Company	Transport
... **Barrel** Racer Turn And Burn T Shirt Trucker Cap ...	barrel	Weapon	NotWeapon

Table 3: List of examples on which the phrasal rules fire with hard match resulting in incorrect labels. Soft-match using our proposed SoftCompose **method restricts these rules and assigns the correct labels.**

Example	Rule phrase	ML prediction (incorrect)	SoftCompose prediction (correct)
... Online-Shopping Buy **bushmaster-223** columbus ...	ar15	NotWeapon	Weapon
... **cannabidiol** legal | Lose 50lbs in 61 Days On Keto ...	vaping	NotTobacco	Tobacco
... **cannibus** seeds ... Is The Newest Place to Search..	vaping	NotTobacco	Tobacco
... help you find **psychoactive** plants - try it now ...	smoke	NotDrugs	Drugs
... Find Relevant Results For **marajuana** seeds. Searching Smarter with Us ...	smoke	NotDrugs	Drugs
... myprotein impact whey **protein** review | Search on our website ...	cod	NotSupplements	Supplements
... Coffee **nicotine** pouches Swedish Pouches Free shipping to Slovenia ...	tobacco	NotTobacco	Tobacco
... **casino** Miami Lakes Experiences, More Gaming 140,000 Sq.Ft of Fun..	black jack	NotGambling	Gambling

Table 4: Generalization using soft-match: List of examples on which the phrasal rules do not fire, and ML model predicts incorrectly. These examples are classified correctly using our soft-match using our proposed SoftCompose **method.**

such work below). we are not aware of any prior work on online adaptation of neural models with symbolic rules.

Neuro-symbolic methods: Unlike our method that seeks online adaptation without updating the neural model, most prior neuro-symbolic methods focus on training a neural model using rules as additional supervision. We review them briefly. A common theme is to view rules as prior knowledge, use them to weakly label unlabeled data and distill such weak supervision to train the neural model. One of the earliest such work is Hu et al. [5] that distills labels induced from hard matches from rules to the neural network using an iterative method. Awasthi et al. [3] et.al. propose to jointly learn to restrict over-generalized rules and the neural model using a soft implication loss between the classifier and rule matching network. On similar lines, Pryzant et al. [9] use rules to fine-tune pre-trained transformer models. Zhang et al. [17] proposes to generate complementary weak labels and re-train the ML model. Seo et al. [13] proposes to learn representations of a fixed number of rules specified as prior knowledge to be used alongside a co-trained neural model. As such these methods are not applicable to our setting of fast online incorporation of feedback rules without modifying the ML classifier.

Online adaptation of neural models: Existing works on fast online adaptation of a trained neural models [11, 14, 15] assume that the feedback is provided as labels to instances, and not as rules. A common strategy in such cases is to store the feedback instances as memory and assign labels to future instances based on similarity with the memory instances and prediction from the base model. A recent state of the art method in this category is the KNNMT algorithm proposed in Khandelwal et al. [6] that treats the memory as a K-nearest neighbor classifier and the model as another classifier. Final prediction is based on a convex combination of the two. While our overall framework is similar, we differ in two critical ways: (1) we design a novel matching algorithm to soften match of symbolic rules with input text and (2) we design rule-specific parameters to handle the diversity of quality in user-provided feedback rules.

6 CONCLUSION

Though ML classifiers are commonly used to decide if advertisements are safe to display in compliance with legal regulations, they make misclassifications during deployment due to distributional shifts, violating compliance. Current industrial practice is to address these violations by writing symbolic feedback rules, and using these rules to override the decisions made by the ML classifier in a "hard" manner. Due to difficulty in writing feedback rules at the right level of precision and generality, such rules are inherently noisy in practice. Consequently, hard composition of ML classifiers with such feedback rules results in poor accuracy as well as compliance. In this paper, we proposed a novel "soft" compliance approach, which generalizes the rules using contextual and non-contextual embeddings, while learning two parameters per rule to decide when to override the output of the ML classifier with the feedback rules. Furthermore, we show how to learn the soft composition operator in an online manner, as new feedback rules arrive incrementally during deployment. We demonstrate that such a soft composition operator significantly improves both accuracy and compliance using industrial advertisement datasets, as well as three publicly available datasets. Future work could try to enhance the expressibility of rules beyond bag of keywords, and design interfaces to help administrators write more accurate rules to start with.

REFERENCES

[1] Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156* (2020).

[2] Josh Attenberg and Foster Provost. 2010. Why Label When You Can Search? Alternatives to Active Learning for Applying Human Resources to Build Classification Models under Extreme Class Imbalance. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 423–432.

[3] Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. *arXiv preprint arXiv:2004.06025* (2020).

[4] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. 2004. Adversarial Classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. Association for Computing Machinery, New York, NY, USA, 99–108.

[5] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[6] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710* (2020).

[7] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

[8] Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1* (Taipei, Taiwan) *(COLING '02)*. Association for Computational Linguistics, USA, 1–7. https://doi.org/10.3115/1072228.1072378

[9] Reid Pryzant, Ziyi Yang, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2022. Automatic Rule Induction for Efficient Semi-Supervised Learning. *arXiv preprint arXiv:2205.09067* (2022).

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

[11] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *ICML*. 1842–1850.

[12] D. Sculley, Matthew Eric Otey, Michael Pohl, Bridget Spitznagel, John Hainsworth, and Yunkai Zhou. 2011. Detecting Adversarial Advertisements in the Wild. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. Association for Computing Machinery, New York, NY, USA, 274–282.

[13] Sungyong Seo, Sercan Arik, Jinsung Yoon, Xiang Zhang, Kihyuk Sohn, and Tomas Pfister. 2021. Controlling Neural Networks with Rule Representations. *Advances in Neural Information Processing Systems* 34 (2021).

[14] Shiv Shankar and Sunita Sarawagi. 2018. Labeled Memory Networks for Online Model Adaptation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 494, 8 pages.

[15] J. Snell, K. Swersky, and R. Zemel. 2017. Prototypical Networks for Few-shot Learning. *ArXiv eprints* (2017). arXiv:1703.05175

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

[17] Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. PRBoost: Prompt-Based Rule Discovery and Boosting for Interactive Weakly-Supervised Learning. *arXiv preprint arXiv:2203.09735* (2022).

[18] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

	Learning-rate	Batch-size	Training steps
AdsDating	2e−5	64	10,000
AdsAddiction	2e−5	64	10,000
AdsHealth	2e−5	64	10,000
AdsAdultWeapon	2e−5	64	10,000
Yahoo	2e−5	64	20,000
TREC	3e−4	16	300
DBpedia	2e−5	64	10,000

Table 5: Hyperparameters for fine-tuning base ML classifier

	Learning-rate	Epoch	k
AdsDating	1, 1e−1, 1e−2	10, 100	0, 64, 512, 1000
AdsAddiction	1, 1e−1, 1e−2	10, 100	0, 64, 512, 1000
AdsHealth	1, 1e−1, 1e−2	10, 100	0, 64, 512, 1000
AdsAdultWeapon	1, 1e−1, 1e−2	10, 100	0, 64, 512, 1000
Yahoo	3e−2, 1e−2	[50, 100]	1000
TREC	1e−2, 5e−3, 2e−3	100, 200, 500	[10, 30, 50]
DBpedia	1, 1e−1, 1e−2	10, 100	0, 64, 512, 1000

Table 6: Grid-search hyperparameters for SoftCompose

	Temperature	Weight
AdsDating	1, 10, 100	0.3, 0.5, 0.7
AdsAddiction	1, 10, 100	0.3, 0.5, 0.7
AdsHealth	1, 10, 100	0.3, 0.5, 0.7
AdsAdultWeapon	1, 10, 100	0.3, 0.5, 0.7
Yahoo	1, 10, 100	0.3, 0.5, 0.7
TREC	1, 10, 100	0.3, 0.5, 0.7
DBpedia	1, 10, 100	0.3, 0.5, 0.7

Table 7: Grid search hyperparameters for KNNMT

A APPENDIX

Appendix is organized as follows. Section A.1 provides details on the hyperparameters used in the experiments. Section A.2 provides more details on the ablation study related to rule precision.

A.1 Hyperparameters

The hyperparameters for fine-tuning the base ML classifier are given in Table 5 for all the datasets. These are taken from the state-of-the-art methods on all the datasets, with the number of training steps chosen so that the accuracy on validation set converges.

The SoftCompose method has the following hyperparameters: (1) learning-rate (2) epochs (3) k (nearest neighbors of rule to select from train set). We tuned these hyperparameters with the grid-search method using the validation compliance error metric and reported the results based on the test set. The KNNMT method uses the following parameters: (1) temperature (2) weight (for knn part). The hyperparameter grid search ranges for both the method are given in Table 6 and Table 7. For averaging the results, we train the system across 5 different shift-unshift partitions and the same hyperparameter sets were used across all the runs.

Datasets	Precision Ranges	0.6 - 0.7		0.8 - 0.9	
	Models	Compliance	Accuracy	Compliance	Accuracy
AdsAddiction	HardCompose	15.95	72.10	3.61	86.74
	SoftCompose	4.22	88.75	2.03	88.60
AdsHealth	HardCompose	14.18	72.06	2.55	86.31
	SoftCompose	1.80	88.35	0.95	88.43
AdsAdultWeapon	HardCompose	13.54	75.67	4.2	87.72
	SoftCompose	3.27	89.28	2.75	89.35
Yahoo	HardCompose	5.92	66.09	1.64	68.83
	SoftCompose	2.39	69.37	1.04	69.56
DBpedia	HardCompose	9.39	86.19	2.60	94.1
	SoftCompose	0.7	96.67	0.39	96.85

Table 8: Comparison of SoftCompose **method with rules from different precision ranges, in terms of compliance and accuracy. We observe that the gains on lower precision ranges is higher than with highly precise rules.**

A.2 Accuracy in different precision ranges

This section presents the compliance and accuracy numbers for the ablation study detailed in Section 4.7.1, We see similar trends in both compliance and accuracy metrics as we vary the rule precision. The gains with highly precise rules are less as compared to the lower precise ones. This is as expected as highly precise rules result in reduced number of conflicts.