Abstract

Extreme Multi-label Text Classification (XMC) involves learning a classifier that can assign an input with a subset of most relevant labels from millions of label choices. Recent works in this domain are increasingly focusing on the problem setting with (i) short-text input data, and (ii) labels endowed with meta-data in the form of textual descriptions. Short-text XMC with label features has found numerous applications in areas such as prediction of Related Searches, product recommendation based on titles, and bid-phrase suggestion, amongst others.

In this work, by exploiting the problem characteristics of short-text XMC, we develop postulates stating the desired invariances, and propose two data augmentation techniques to achieve them. One, LabelMix, which performs data augmentation by concatenating an annotating label to the data-point; and the other, Gandalf, which generates additional data-points by considering labels as legitimate data-points. The efficacy of the proposed augmentation methods is demonstrated by showing upto 30% relative improvement when applied to a range of existing algorithms, and proposing an algorithmic framework, InceptionXML-LF, which furthers state-of-the-art on benchmark datasets.

1 Introduction

Related Searches, product recommendation and bid-phrase matching tasks require predicting the most relevant results that are either highly correlated or frequently co-occur with the given input query/product. Extreme Multilabel Classification (XMC) has found multiple applications in these domains where the problem is modelled as a short-text classification task over millions of possible searches/products/ad-phrases considered as labels. Real world data from these domains, when modelled as an XMC problem, is highly imbalanced towards some popular or trending ad-phrases/products and notoriously exhibits fit to Zipf’s law. Here, most labels in the extremely large output space, are tail labels i.e, those which have very few ($\leq$ 5) instances in the training set (Babbar and Schölkopf, 2019). Similarly, the words in queries also follow a long-tailed distribution. While there exists insufficient training data for these tail labels/words, the short-text nature of these queries makes it no simpler for the models to learn meaningful, non-overfitting embeddings and encoded representations for tail words and labels.

Due to the increasing requirement of scalable and low latency models in these domains, there has been a surge in works that model recommendations tasks like related searches, query-to-product and document-to-document recommendation as a short-text XMC problem using only the search query, product name or document title. Hence, most of these works are focused on building lightweight and frugal architectures that can predict in milliseconds and scale up to millions of labels. Despite being frugal in terms of number of layers/parameters in the network, these models can learn the training data well enough. Hence, creating deeper models for better representation learning is perhaps not the most optimal solution under this setting.

Many of the recent works, thus, make architectural improvements to leverage “label features” in order to imbue strong inductive biases in their models. A label feature is the text associated with labels, which spans the same vocabulary universe as query text. Label features, when encoded in the same embedding space as query texts, help mitigate the difficulty in learning efficient representations for tail labels by enabling joint query-label representation learning in common embedding space (Dahiya et al., 2021a; Mittal et al., 2021a,b); thereby improving the prediction performance. However, even after improved representation learning by leveraging label features, a significant generalization gap is noticeable in these approaches (Figure 4, Appendix B).
Contributions In this work, we take a data-centric approach, and aim at answering “Can we extend mixup to feature-label extrapolation to guarantee a robust model behavior far away from the training data?”, a question posed in Zhang et al. (2018). To this end, we (i) propose LabelMix augmentation and motivate it through the Vicinal Risk Minimization (VRM) (Chapelle et al., 2000) principle, which is achieved by explicating the desired invariance properties and leveraging them for augmentation purposes. To the best of our knowledge, this is the first work that attempts to further the application of vicinal risk minimization to an embedding space where data instances and their label features co-exist in a shared embedding space.

We further (ii) discuss self and soft-annotation properties of label features and propose Gandalf - GrAPh inDuced Data Augmentation based on Label Features - to efficiently leverage label features as valid training instances. (iii) As an algorithmic contribution, we propose an extension to INCEPTIONXML (Kharbanda et al., 2021), to accommodate label features, as an efficient alternative framework to current short-text XMC pipelines (See Appendix A). (iv) We demonstrate the generality and effectiveness of the proposed data augmentations, by showing upto 30% relative improvements in multiple state-of-the-art extreme classifiers on public benchmark datasets. Our experiments reflect that strong inductive biases that are currently imbuied into models through complicated training pipelines and architectural modifications can also be induced with simple data augmentation techniques as proposed in the paper.

2 Related Work

Earlier works in XMC have focused on the problem of tagging long text documents consisting of hundreds of tokens. These are broadly categorized based on their algorithmic characteristics as follows: (i) Label-tree methods (Jasinska et al., 2016; Prabhul et al., 2018; Khandagale et al., 2020), (ii) Decision tree-based methods (Prabhu and Varma, 2014; Choromanska and Langford, 2015; Agrawal et al., 2013) (iii) Label-embedding methods (Bhattia et al., 2015; Yu et al., 2014; Tagami, 2017), (iv) One-vs-rest methods (Babbar and Schölkopf, 2017; Yen et al., 2017), and (v) Deep learning methods (Liu et al., 2017; You et al., 2019). Of late, works aimed towards scaling up transformer encoders for XMC have dominated the research landscape in this domain (Chang et al., 2020; Ye et al., 2020; Zhang et al., 2021).

XMC with Label Features: More recent works have shifted their focus to short-text XMC to keep up with the increasing requirements of low-latency models in recommendation tasks. These works can be split into two categories: (i) those which inherently do not leverage label features like ASTEC (Dahiya et al., 2021b) and INCEPTIONXML (Kharbanda et al., 2021), and (ii) those which do heavy architectural modifications or employ complicated training strategies in order to leverage label features along with short-text instances to induce strong inductive bias into their models. For example, SIAMESEXML (Dahiya et al., 2021a) employs a siamese constrastive learning stage between instance and its label features through a modified negative log-likelihood loss and, DECAF (Mittal et al., 2021a) and ECLARE (Mittal et al., 2021b) extend the DEEPXML (Dahiya et al., 2021b) pipeline by augmenting the ASTEC encoder with one or two extra ASTEC-like encoders for label-text and graph-augmented label text and as a result end up taking ~3x the time as compared to ASTEC.

Data Augmentation Apart from the architectural design choices, data augmentation methodologies have been successful in providing much needed inductive biases leading to better performance of machine learning models on unseen data. While these have been highly popular for computer vision tasks inspired by recent works (Zhang et al., 2018; Verma et al., 2019; Yun et al., 2019), augmentation techniques remain relatively under-explored in Natural Language Processing. However, recent works have shown their efficacy in NLP tasks such as machine translation (Gao et al., 2019), common-sense reasoning (Yang et al., 2020), semantic parsing (Guo et al., 2020), text classification (Zhao et al., 2022; Wei and Zou, 2019), and for achieving adversarial robustness (Li et al., 2017). Further details on tasks specific techniques for data augmentation in NLP can be found in Feng et al. (2021). For (short-text) XMC, which is the focus of this paper, there have been no works which have leveraged data augmentation or studied its implications.

3 Background & Notation

For training, we have available a dataset \( \mathcal{D} = \{\{x_i, y_i\}_{i=1}^N, \{z_i\}_{i=1}^L\} \) of \( N \) pairs of input datapoints \( x_i \), their corresponding labels \( y_i \), and a label...
Table 1: Characteristics of short-text benchmark datasets with label features. Here, APPL stands for avg. points per label, ALpP stands for avg. labels per point and AWpP is length i.e. avg. words per point.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>N</th>
<th>L</th>
<th>ALpP</th>
<th>AWpP</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF-AmazonTitles-131K</td>
<td>294,805</td>
<td>151,073</td>
<td>5.15</td>
<td>2.29</td>
</tr>
<tr>
<td>LF-WikiSeeAlsoTitles-320K</td>
<td>693,082</td>
<td>312,330</td>
<td>4.67</td>
<td>2.11</td>
</tr>
<tr>
<td>LF-WikiTitles-300k</td>
<td>1,813,391</td>
<td>501,070</td>
<td>17.15</td>
<td>4.74</td>
</tr>
</tbody>
</table>

In the XMC pipelines is via the one-vs-all (OVA) scheme, such that for each $x_i$, a score $s_l(x_i)$ is calculated for each label $l \in [L]$. In practice, the sum over all labels is very expensive, and is therefore often approximated e.g. by a label shortlisting procedure (Jain et al., 2019; Jiang et al., 2021).

In the OVA paradigm, scores are typically calculated by projecting both the instances and the labels to some common Euclidean space $\mathcal{E} = \mathbb{R}^d$, and then taking their inner product. The mapping of instances to embeddings is realized through a feature extractor $\Phi : \mathcal{X} \rightarrow \mathcal{E}$. All labels are decoded through $\Psi \equiv \{w_i\}_{i=1}^L$, where $w_i \in \mathcal{E}$ is the label’s decoding representation. In this notation, we have $s_l(x) = \langle \Phi(x), \Psi(l) \rangle$.

The short-text XMC problem is characterized by two random variables $X \in \mathcal{X}$ and $Y \in \{0, 1\}^L$ jointly distributed according to $\mathbf{P}$. Here, the labels are sparse, $\mathbb{E}[||Y||_1] = \bar{y} \ll L$, and follow a long-tailed distribution. Also, the instances are short-text i.e. $\mathbb{E}[\text{len}(X)] = \bar{m}$, with $\bar{m}$ in the range of about 3 to 8 tokens\(^1\) as shown in Table 1.

\(^1\)We do not place a strict upper-bound on the number of tokens, because this complicates the concatenation arguments

4 Invariances in Short-text XMC & Vicinal Risk Minimization

The extreme scarcity of training data for tail labels in XMC implies that a good classifier for these labels can only be learned if, in addition to the few training examples, strong inductive biases are employed during training. Even though there exist problem-agnostic regularizations such as limiting the magnitude of the parameters (via $L_1$ or $L_2$ regularization), implicit regularization through SGD dynamics, or dropout, it is beneficial to use domain knowledge for more efficient inductive biases.

As discussed in section 2, many recent XMC baselines leverage label features in order to imbue strong inductive biases in their models either through computationally expensive architectural additions or complicated training procedures. If similar inductive biases could be achieved through data augmentation, then these would not be restricted to a single architecture, but benefit most current and future short-text XMC methods. Thus, our goal is to identify underlying properties of the short-text XMC problem, and use these to derive new data augmentation techniques.

Data augmentation can be seen as a form of vicinal risk minimization, the idea that one minimizes risk over the empirical distribution

$$\mathbb{D}_P(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}(x) \delta_{y_i}(y),$$

but instead over a smoothed out version $\mathbb{P}_\nu$. That means that each data point $x$ in the input, corresponding to a peak $\delta_x$ in the empirical distribution, instead is turned into a smooth distribution that has nonzero density in the vicinity of $x$. The key task is then to determine what constitutes the vicinity of a data point in this setting.

Symmetries and transformation laws have long been a fruitful source for inductive biases in machine learning. For example, in computer vision the underlying symmetries are, for example, translation, rotation, and flips. Invariance under small translations is typically achieved through the network architecture, by using convolution layers which are covariant\(^2\) and pooling layers which are invariant to small shifts. For more complex trans-

\(^2\)A note on terminology: We categorize possible transformation behaviour into three groups: Covariant (sometimes called equivalent), where the output transforms in the same manner as the input, contravariant, where the output transforms in the opposite way as the input, and invariant, where the transformation in the input leaves the output unchanged.
formations, such as the rotations and flips, however, data augmentation is required. For continuous transformations, like rotations, translations, or scaling, one can easily postulate that the classification should remain invariant if the change is very small.

Due to the discrete nature of text input in language tasks, however, there are no continuous transformations available. It has been shown in recent works that simple discrete transformations of input data such as replacement with synonym, introduction of typos, and swapping of neighboring texts can improve the performance to a certain extent (Xie et al., 2017; Coulombe, 2018; Wei and Zou, 2019; Niu and Bansal, 2018), however, and such transformations can lead to semantic inconsistency and illegibility. Therefore, we have to look deeper into the actual properties of the short-text data to find transformations with predictable behaviour.

**Considerations for input concatenations in short-text XMC**

For textual data, combining data-points via direct concatenation of input texts of other data-points, can lead to significant changes in their intended meaning. In such a setting, one might assume that if two inputs are joined together, the resulting labels would be the union of the labels of the two data points. Especially for longer text, this could be seen as a sensible approach, e.g. if a Wikipedia article consists of two sections, then the tags for that article could be the union of the tags for each section. This can be encoded as follows:

**Hypothesis 1 (Concatenation Covariance).** For two (long-text) inputs \((x_i, y_i)\) and \((x_j, y_j)\), where \(y_i, y_j \subseteq \mathcal{L}\) are represented as sets, concatenation of inputs corresponds to union of sets

\[\Phi(x_i \oplus x_j) = y_i \cup y_j.\]  

(2)

However, for short text, one could argue that the opposite is true. If a user adds additional words to a search query, a Wikipedia page, or a product name, then these words are often meant to filter the results further. For example, changing the search query from “Boat Wireless headphones” to “Boat Wireless headphones with microphone” would lead to a filtered result. This leads to the opposite hypothesis

**Hypothesis 2 (Concatenation Contravariance).** For two input queries \((x_i, y_i)\) and \((x_j, y_j)\), where
$y_i, y_j \subset [L]$ are represented as sets, concatenation of inputs corresponds to intersection of sets
\[ \Phi(x_i \oplus x_j) = y_i \cap y_j. \] (3)

Further, we have to concede that for two arbitrary short-text queries/product names, it is very difficult to predict the exact meaning of their concatenation. One could argue to use combine the queries in the manifold space (Verma et al., 2019), but as shown in Figure 4, while Manifold Mixup does help reduce the overfitting, it still does not imbue enough inductive bias into the model.

In the case where label features are also short-text data, we can at least identify a subset of concatenations that should leave the classification unchanged. That is, if the second input text can be known beforehand to only reaffirm the content of the first text. Therefore, choosing the second text to be one of the relevant label’s text should result in an invariance. From examples in section 5 and Figure 5, one can observe that concatenating a query with one of its label feature only reaffirms the content of the query. This brings us to

**Postulate 1 (Label-Affirming Concatenations).**

Let $(x, y)$ be a training data point in $\mathcal{D}$, and $j \in y$ be a label relevant to $x$. Then the classifier should be invariant under concatenation with $z_j$
\[ \Phi(x \oplus z_j) = \Phi(x). \] (4)

This is corroborated by Figure 2, where queries concatenated with label features in the input space $\Phi(x \oplus z_i)$ have their encoded representations in the vicinity (indicated by high cosine similarity) of the encoded representation of only queries as input $\Phi(x)$. Thus, this postulate enables us to specify the vicinal distribution. Given a datapoint $(x, y) \in \mathcal{D}$, its vicinity is given by $V(x) := \{x \oplus z_j : j \in y\}$.

The straightforward way to define the vicinal distribution would be to sample uniformly on $V(x)$. However, as the main goal of the augmentation is to improve the generalization on tail labels, it can be beneficial to allow for weighted distributions. Using an instance-independent weight vector $r \in \mathbb{R}^L$, the probability of choosing $x \oplus z_j$ as the augmented label is given by $y_j r_j / \langle y_i, r \rangle$, where the first term ensures that $j$ is actually a relevant label, the second term is the weighting factor, and the third the normalization. Averaging over the entire dataset thus leads to the vicinal distribution:
\[ d\mathcal{P}_v(x, y) = \frac{1}{n} \sum_{i=1}^{n} \frac{\delta_{y_i}(y)}{\langle y_i, r \rangle} \sum_{j=1}^{L} y_j r_j \delta_{x_i \oplus z_j}(x). \]
The above postulate suggests a methodology to use label features as data instances with the self-annotation property. One natural question arises regarding the labels for the thus created data instances (i.e., via self-annotation): In a label space $[L]$ comprising of hundreds of thousands or millions of labels, what are the suitable labels $l' \neq l$ for $z_l$, when posed as a data instance? According to the observation (i) in Section 5, the labels that are highly correlated to or the ones that frequently co-occur with a label $l$ should, ideally, also make up as the label-set for the label feature $z_l$, when searched as a query or posed as a data instance.

One way to approximate label correlations or co-occurrences is to use the Label Correlation Graph, as proposed in ECLARE, which gives a smooth approximation of label-occurrences purposely skewed in favor of tail labels. Since the entries in LCG are normalized, these can be interpreted regularized variants of the label co-occurrence matrix. As argued in ECLARE, for each label, the LCG finds a set of semantically similar labels that either share tokens with the label, or are used in the same context. This can be further seen in Figure 5 where the degree of correlation of a label with its first order neighbours in the LCG has been plotted. While ECLARE uses the LCG efficiently to either re-weight logits as a post-processing step, or to augment labels’ decoding representations $w_i$, we propose to leverage the graph weights (with an additional row-wise normalization to get values in range $[0, 1]$) as probabilistic soft labels for $z_l$ as input data instance. We thereby propose to use the following postulate:

**Postulate 3 (Soft-annotations via LCG).** If the features $z_l$ of a label are interpreted as a data instance; then beyond self-annotation, $z_l$ can be soft-annotated with the other labels $l' \neq l$ either highly correlated to or that frequently co-occur with $l$. Therefore, we require the following:

$$P[Y_{l'} = 1 | X = z_l] = LCG[l, l']$$  

### 6 Proposed Augmentations in Practice

**LabelMix Augmentation** Through LabelMix, we propose a way of performing label-affirming concatenations where, a data point is chosen with a uniform probability of 0.4 and is concatenated with one of it’s label features and the corresponding label space is formed by the concatenation of the
we assign the labels weights according to a skewed
word embeddings are frozen.
given their multi-stage pipelines where, in some stages, the
LabelMix
350K, and LF-WikiTitles-500K. We test the gener-
mark our experiments on 3 standard public datasets
We bench-
Benchmarks, Baselines & Metrics : We benchmark our experiments on 3 standard public datasets
and Gandalf augmentations across multiple state-
of-the-art short-text extreme classifiers, the de-
tails of which have been discussed in section 2.
Similarly, to test the effectiveness of the INCEPTIONXML encoder on these datasets, we extend the model to leverage label features and call it INCEPTIONXML-LF. For this, we augment it with additional label-text and graph-augmented label text classifiers as done in Mittal (2021b). We further create a 2-stage training strategy as compared to a single one in INCEPTIONXML. The implementation details and training strategy can be found in Appendix A. We measure the models’ performance using standard metrics precision@k, denoted P@k, and its propensity-scored version PSP@k (Jain et al., 2016).

7.1 Results
We can make some key observations and develop strong insights not only about the short-text XMC problem with label features but also about specific dataset properties from Table 2. For example, while the effect of data augmentations can be observed strongly in the first two datasets, only limited improvement are noticeable in LF-WikiTitles-500K dataset. This can be attributed to LF-WikiTitles-500K having three times as many data points, on average, per label (Table 1), and thus not requiring as much inductive bias as the other two datasets.

LabelMix LabelMix produces synthetic data points in the vicinity of training data and hence, while being effective in capturing correlations between data instances and tail labels, can only imbue limited additional inductive bias into the model. ECLARE, on the other hand, is able to better capture these correlations through its LCG-augmented classifier and only gains trivially as it only encodes label text in its classifier which leaves out the scope to capture query-tail label correlations further. Similarly, INCEPTIONXML stands to gain significantly more from LabelMix as compared to it’s LF-counterpart which, similar to ECLARE, also employs a LCG-augmented classifier. Notably, LabelMix works much better on INCEPTIONXML-LF because of their dynamic negative mining pipeline, which enables the augmentation to work more effectively.

Gandalf We witness exceptional increase in prediction performance over multiple state-of-the-art extreme classifiers with the Gandalf augmentation,
The best-performing approach is in **bold**.

Table 2: Effect of adding LabelMix and Gandalf data augmentations on state-of-the-art extreme classifiers in terms of P@k and PSP@k public benchmark datasets. The best-performing approach is in **bold**.

<table>
<thead>
<tr>
<th>Method</th>
<th>P@1</th>
<th>P@3</th>
<th>P@5</th>
<th>PSP@1</th>
<th>PSP@3</th>
<th>PSP@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>InceptionXML</td>
<td>32.25</td>
<td>21.70</td>
<td>15.61</td>
<td>23.97</td>
<td>28.60</td>
<td>32.57</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>39.17</td>
<td>26.85</td>
<td>19.49</td>
<td>30.50</td>
<td>36.79</td>
<td>43.95</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>41.42</td>
<td>30.19</td>
<td>21.63</td>
<td>35.80</td>
<td>40.96</td>
<td>46.19</td>
</tr>
<tr>
<td>ASTEC</td>
<td>37.12</td>
<td>25.20</td>
<td>18.24</td>
<td>29.22</td>
<td>34.64</td>
<td>39.49</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>39.75</td>
<td>26.55</td>
<td>18.59</td>
<td>29.91</td>
<td>35.58</td>
<td>40.63</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>43.95</td>
<td>29.66</td>
<td>21.39</td>
<td>37.40</td>
<td>43.03</td>
<td>48.31</td>
</tr>
<tr>
<td>DECAF</td>
<td>38.4</td>
<td>25.84</td>
<td>18.65</td>
<td>30.85</td>
<td>36.44</td>
<td>41.42</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>39.30</td>
<td>26.60</td>
<td>19.23</td>
<td>31.81</td>
<td>37.67</td>
<td>42.83</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>42.43</td>
<td>28.96</td>
<td>20.90</td>
<td>35.22</td>
<td>42.12</td>
<td>47.61</td>
</tr>
<tr>
<td>ECLARE</td>
<td>40.46</td>
<td>27.54</td>
<td>19.63</td>
<td>33.18</td>
<td>39.55</td>
<td>44.10</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>40.34</td>
<td>27.54</td>
<td>19.96</td>
<td>33.48</td>
<td>39.74</td>
<td>45.11</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>42.51</td>
<td>28.89</td>
<td>20.81</td>
<td>35.72</td>
<td>42.19</td>
<td>47.46</td>
</tr>
<tr>
<td>INCEPTIONXML</td>
<td>36.79</td>
<td>24.94</td>
<td>17.05</td>
<td>28.50</td>
<td>34.15</td>
<td>38.79</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>40.41</td>
<td>27.45</td>
<td>19.82</td>
<td>31.22</td>
<td>38.54</td>
<td>43.81</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>44.67</td>
<td>30.00</td>
<td>21.50</td>
<td>37.98</td>
<td>43.83</td>
<td>48.93</td>
</tr>
<tr>
<td>INCEPTIONXML-LF</td>
<td>40.74</td>
<td>27.24</td>
<td>19.57</td>
<td>34.52</td>
<td>39.40</td>
<td>44.13</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>41.90</td>
<td>28.20</td>
<td>20.35</td>
<td>35.60</td>
<td>41.07</td>
<td>46.20</td>
</tr>
<tr>
<td>+ Gandalf</td>
<td>43.84</td>
<td>29.59</td>
<td>21.30</td>
<td>38.22</td>
<td>43.90</td>
<td>49.03</td>
</tr>
</tbody>
</table>

Table 3: Ablation results demonstrating the effectiveness of using soft-annotations (denoted SA) obtained from the LCG on a single InceptionXML model. Notably, soft annotations play an important role in learning label-label correlations.

<table>
<thead>
<tr>
<th>Method</th>
<th>P@1</th>
<th>P@3</th>
<th>P@5</th>
<th>PSP@1</th>
<th>PSP@3</th>
<th>PSP@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabelMix w/o SA</td>
<td>35.62</td>
<td>24.13</td>
<td>17.35</td>
<td>27.53</td>
<td>33.06</td>
<td>37.50</td>
</tr>
<tr>
<td>LabelMix w SA</td>
<td>37.25</td>
<td>25.02</td>
<td>17.98</td>
<td>29.28</td>
<td>34.58</td>
<td>39.09</td>
</tr>
<tr>
<td>InceptionXML</td>
<td>39.05</td>
<td>26.52</td>
<td>19.15</td>
<td>30.98</td>
<td>37.20</td>
<td>42.26</td>
</tr>
<tr>
<td>Gandalf w SA</td>
<td>37.59</td>
<td>25.25</td>
<td>18.18</td>
<td>30.75</td>
<td>35.54</td>
<td>40.06</td>
</tr>
<tr>
<td>+ LabelMix</td>
<td>43.53</td>
<td>29.23</td>
<td>20.92</td>
<td>36.96</td>
<td>42.71</td>
<td>47.64</td>
</tr>
</tbody>
</table>

Table 4(Appendix B): Further visualizations depicting differences in predictions or employ complicated training pipelines to learn strong inductive biases. For instance, ASTEC and InceptionXML beat their LF-counterparts DECAF and ECLARE, and InceptionXML-LF respectively on LF-AmazonTitles-131K, while performing at par with them on other two datasets.

8 Conclusion

In this paper, we proposed two data augmentation methods which are particularly suited for short-text extreme classification. These augmentations not only eliminate the need for complicated training procedures in order to imbue inductive biases, but dramatic increase in prediction performance of state-of-the-art methods in this domain. It is expected that our treatment towards studying invariances in this domain will spur further research.
References


A. Choromanska and J. Langford. 2015. Logarithmic time online multiclass prediction. NIPS.


H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In WSDM.


Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In EMNLP.


H. Yu, P. Jain, P. Kar, and I. S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In ICML.


A INCEPTIONXML-LF

Model Outlook: Short-text queries are encoded by a modified InceptionXML encoder, which encodes an input query $x_t$ using an encoder $\Phi_t := (E, \theta)$ parameterised by $E$ and $\theta$, where $E$ denotes a $D$-dimensional embedding layer of $\mathbb{R}^{V \times D}$ for vocabulary tokens $V = \{t_1,t_2,\ldots,t_V\}$ and $\theta$ denotes the parameters of the embedding enhancement and the inception module respectively. Alongside $\Phi_t$, INCEPTIONXML-LF learns two frugal ASTEC-like (Dahiya et al., 2021b) encoders, one each as a label-text encoder $\Phi_I := \{E,R\}$ and a graph augmented encoder $\Phi_g := \{E,R\}$. Here, $R$ denotes the parameters of a fully connected layer bounded by a spectral norm and the embedding layer $E$ is shared between all $\Phi_t$, $\Phi_I$ and $\Phi_g$ for joint query-word label learning. Further, an attention module $A$, meta-classifier $\mathcal{W}_m$ and an extreme classifier $\mathcal{W}_e$ are also learnt together with the encoders. Next, we specify the details of all components of INCEPTIONXML-LF.

A.1 Instance-Attention in Query Encoder

We make two improvements to the inception module INCEPTIONXML for better efficiency. Firstly, in the inception module, the activation maps from the first convolution layer are concatenated before passing them onto the second convolution layer. To make this more computationally efficient, we replace this “inception-like” setting with a “mixture of expert” setting (Yang et al., 2019). Specifically, a route function is added that produces dynamic weights for each instance to perform a dynamic element-wise weighted sum of activation maps of each filter. Along with the three convolutional experts, we also add an average pool as a down sampling residual connection to ensure better gradient flow across the encoder.

Second, we decouple the second convolution layer to have one each for the meta and extreme classification tasks.

A.2 Dynamic Hard Negative Mining

Training one-vs-all (OvA) label classifiers becomes infeasible in the XMC setting where we have hundreds of thousands or even millions of labels. To mitigate this problem, the final prediction or loss calculation is done on a shortlist of size $\sqrt{L}$ comprising of only hard-negatives label. This mechanism helps reduce complexity of XMC from an intractable $O(NDL)$ to a computationally feasible $O(ND\sqrt{L})$ problem. INCEPTIONXML-LF inherits the synchronized hard negative mining framework as used in the INCEPTIONXML. Specifically, the encoded meta representation is passed through the meta-classifier which predicts the top K relevant label clusters per input query. All labels present in the top-K shortlisted label clusters then form the hard negative label shortlist for the extreme task. This allows for progressively harder labels to get shortlisted per short-text query as the training proceeds and the encoder learns better representations.

A.3 Label-text and LCG Augmented Classifiers

INCEPTIONXML-LF’s extreme classifier weight vectors $W_e$ comprise of 3 weights, as in Mittal et al. (2021b). Specifically, the weight vectors are a result of an attention-based sum of (i) label-text embeddings, created through $\Phi_t$, (ii) graph augmented label embeddings, created through graph encoder $\Phi_g$ and, (iii) randomly initialized per-label independent weights $w_l$. As shown in Figure 3, we first obtain label-text embeddings as $z_l^1 = E \cdot z_l^0$, where $z_l^0$ are the TF-IDF weights of label feature corresponding to label $l$. Next, we use the label correlation graph $G$ to create the graph-weighted label-text embeddings $z_l^2 = \sum_{m \in [l]} G_{lm} \cdot z_m^0$ to capture higher order query-tail label correlations. $z_l^1$ and $z_l^2$ are then passed into the frugal encoders $\Phi_I$ and $\Phi_g$ respectively. These encoders comprise only of a residual connection across a fully connected layer as $\alpha \cdot R \cdot G(\tilde{z}_l) \beta \cdot \tilde{z}_l$, where $\tilde{z}_l = \{z_l^1, z_l^2\}$, $G$ represents GELU activation and $\alpha$ and $\beta$ are learned weights. Finally, the per-label weight vectors for the extreme task are obtained as

$$W_{e,l} = A(z_l^1, z_l^2, w_l) = \alpha^1 \cdot z_l^1 + \alpha^2 \cdot z_l^2 + \alpha^3 \cdot w_l$$

where $A$ is the attention block and $\alpha^{1,2,3}$ are the dynamic attention weights produced by the attention block.

A.4 Two-phased Training

Motivation: We find there to be a mismatch in the training objectives in DeepXML-based approaches like ASTEC, DECAF and ECLARE which first train their word embeddings on meta-labels in Phase I and then transfer these learnt embeddings for classification over extreme fine-grained labels in Phase III (Dahiya et al., 2021b). Thus,
in our two-phased training for INCEPTIONXML-LF, we keep our training objective same for both phases. Note that, in INCEPTIONXML-LF the word embeddings are always learnt on labels instead of meta-labels or label clusters and we only augment our extreme classifier weight vectors $W_e$ with label-text embeddings and LCG weighted label embeddings. We keep the meta-classifier $W_m$ as a standard randomly initialized classification layer.

**Phase I:** In the first phase, we initialize the embedding layer $E$ with pre-trained GloVe embeddings (Pennington et al., 2014), the residual layer $R$ in $\Phi_l$ and $\Phi_g$ is initialized to identity and the rest of the model comprising of $\Phi_q$, $W_m$ and $A$ is randomly initialized. The model is then trained end-to-end but without using free weight vectors $w_l$ in the extreme classifier $W_e$. This set up implies that $W_e$ only consists of weights tied to $E$ through $\Phi_l$ and $\Phi_g$ which allows for efficient joint learning of query-label word embeddings (Mittal et al., 2021a) in the absence of free weight vectors. Model training in this phase follows the INCEPTIONXML+ pipeline as described in Kharbanda et al. (2021) without detaching any gradients to the extreme classifier for the first few epochs. In this phase, the final per-label score is given by:

$$P_l = A(\Phi_l(z^1_l), \Phi_g(z^2_l), w_l) \cdot \Phi_q(x)$$

**Phase II:** In this phase, we first refine our clusters based on the jointly learnt word embeddings. Specifically, we recluster the labels using the dense $z^1_l$ representations instead of using their sparse PIFA representations (Chang et al., 2020) and consequently reinitialize $W_m$. We repeat the Phase I training, but this time the formulation of $W_e$ also includes $w_l$ which are initialised with the updated $z^1_l$ as well. Here, the final per-label score is given by:

$$P_l = A(\Phi_l(z^1_l), \Phi_g(z^2_l), w_l) \cdot \Phi_q(x)$$

**B Visualizations and Extra Results**

Additional visualizations capturing the label correlations and their first order-neighbors are shown in Figure 5. The relative comparison of outputs generated by vanilla model, and those as a result of the proposed augmentations is shown in Table 3.

**C Limitations**

Our work is limited to the extreme classification problem setting in which the labels are endowed with textual descriptions, and the input data-points are short-text instances as those encountered in Search and recommendation problems based on product titles.
Figure 4: Effect of different data augmentations on InceptionXML-LF. Remarkable improvements can be noted as a result of using the proposed data augmentations LabelMix and Gandalf. However, from (a) and (b), a significant generalization can be observed between Train and Test P@1. While manifold mixup is effective in reducing overfitting, it only makes trivial improvements to the prediction performance.

Figure 5: Correlations between labels and their first-order neighbours, as found by the LCG on the LF-WikiTitles-500K dataset. The legend shows the label in question, the bar chart shows the degree of correlation with its neighbouring labels. Correlated labels often share tokens with each other and/or may be used in the same context.
<table>
<thead>
<tr>
<th>Method</th>
<th>Datapoint</th>
<th>Vanilla Predictions</th>
<th>LabelMix Predictions</th>
<th>Gandalf Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InceptionXML-LF</strong></td>
<td>Topological group</td>
<td>Topological order, Algebraic group, Topological quantum field theory, Topological quantum number, Quantum topology</td>
<td>Topological order, Algebraic group, Topological quantum field theory, Topological quantum number</td>
<td>Compact group, Haar measure, Lie group, Algebraic group, Topological ring</td>
</tr>
<tr>
<td><strong>Decaf</strong></td>
<td>Topological quantum computer, Topological order, Topological quantum field theory, Topological quantum number, Quantum topology</td>
<td>Topological order, Algebraic group, Topological quantum field theory, Topological quantum number</td>
<td>Topological order, Algebraic group, Topological quantum field theory, Topological quantum number</td>
<td>Compact group, Topological order, Lie group, Algebraic group, Topological ring</td>
</tr>
<tr>
<td><strong>Eclaire</strong></td>
<td>Topological quantum computer, Topological order, Topological quantum field theory, Topological quantum number, Quantum topology</td>
<td>Topological order, Topological ring, Topological quantum field theory, Topological quantum number</td>
<td>Topological order, Topological ring, Topological quantum field theory, Topological quantum number</td>
<td>Compact group, Topological order, Lie group, Algebraic group, Topological ring</td>
</tr>
</tbody>
</table>

| **InceptionXML-LF** | Out                              | Oatcake, Oat milk, Rolled oats, List of oat diseases, Goboat | Oatcake, Oat milk, Rolled oats, List of oat diseases, Goboat | Oatcake, Oat milk, Rolled oats, List of oat diseases, Goboat |
| **Decaf**   | Out                              | Oatcake, Oatmeal, Design for All, Oatley Point Reserve, Oatley Pleasure Grounds         | Oatcake, Oatmeal, Design for All, Oatley Point Reserve, Oatley Pleasure Grounds         | Oatcake, Oatmeal, Design for All, Oatley Point Reserve, Oatley Pleasure Grounds |
| **Eclaire** | Oatmeal, Oat milk, Parks in Sydney, Oatley Point Reserve, Oatley Pleasure Grounds | Oatmeal, Oatmeal, Oat milk, Oatley Point Reserve, Oatley Pleasure Grounds | Oatmeal, Oatmeal, Oat milk, Oatley Point Reserve, Oatley Pleasure Grounds | Oatmeal, Oatmeal, Oat milk, Oatley Point Reserve, Oatley Pleasure Grounds |

| **Decaf**   | Grand Lake, Colorado              | Front Range Urban Corridor, Index of Colorado-related articles, National Register of Historic Places listings in Grand County, Colorado | Front Range Urban Corridor, Index of Colorado-related articles, National Register of Historic Places listings in Grand County, Colorado | Front Range Urban Corridor, Index of Colorado-related articles, National Register of Historic Places listings in Grand County, Colorado |
| **Eclaire** | State of Colorado, Colorado cities and towns, Colorado metropolitan areas, Grand County, Colorado, Grand County, Colorado | Colorado metropolitan areas, State of Colorado, Colorado cities and towns, Colorado metropolitan areas, Grand County, Colorado, Grand County, Colorado | Colorado metropolitan areas, State of Colorado, Colorado cities and towns, Colorado metropolitan areas, Grand County, Colorado, Grand County, Colorado | Colorado metropolitan areas, State of Colorado, Colorado cities and towns, Colorado metropolitan areas, Grand County, Colorado, Grand County, Colorado |


| **Eclaire** | Armed Forces of Saudi Arabia      | Saudi-led intervention in Yemen, Saudi-led intervention in Bahrain, Human rights in Saudi Arabia, Legal system of Saudi Arabia, Joint Chiefs of Staff | Saudi-led intervention in Yemen, Saudi-led intervention in Bahrain, Human rights in Saudi Arabia, Legal system of Saudi Arabia, Joint Chiefs of Staff | Saudi-led intervention in Yemen, Saudi-led intervention in Bahrain, Human rights in Saudi Arabia, Legal system of Saudi Arabia, Joint Chiefs of Staff |

| **InceptionXML-LF** | List of armed groups in the Syrian Civil War, Military of Saudi Arabia | Military of Saudi Arabia | Military of Saudi Arabia | Military of Saudi Arabia |

Table 4: Prediction examples of different datapoints from the LF-WikiSeeAlsoTitles-320K dataset. Labels indicate mispredictions. It may be noted that queries with even just a single word, like "Oat", which has random labels in the case of a Vanilla Prediction, gets all the labels right with the addition of Gandalf. Furthermore, even mispredictions get closer when our data augmentation strategy is introduced.