Vec2Node: Self-training with Tensor Augmentation for Text Classification with Few Labels

Sara Abdali[1⋆], Subhabrata Mukherjee[2], and Evangelos E. Papalexakis[3]

[1] Georgia Institute of Technology sabdali3@gatech.edu
[2] Microsoft Research Subhabrata.Mukherjee@microsoft.com
[3] University of California, Riverside epapalex@cs.ucr.edu

Abstract. Recent advances in state-of-the-art machine learning models like deep neural networks heavily rely on large amounts of labeled training data which is difficult to obtain for many applications. To address label scarcity, recent work has focused on data augmentation techniques to create synthetic training data. In this work, we propose a novel approach of data augmentation leveraging tensor decomposition to generate synthetic samples by exploiting local and global information in text and reducing concept drift. We develop `Vec2Node` that leverages self-training from in-domain unlabeled data augmented with tensorized word embeddings that significantly improves over state-of-the-art models, particularly in low-resource settings. For instance, with only 1% of labeled training data, `Vec2Node` improves the accuracy of a base model by 16.7%. Furthermore, `Vec2Node` generates explicable augmented data leveraging tensor embeddings.

Keywords: Text Augmentation · Tensor Decomposition · Self-training

1 Introduction

In recent years, neural network models have obtained state-of-the-art performance in several language understanding tasks employing non-contextualized `FastText` [4] as well as contextualized `BERT` [5] word embeddings. Even though these models have been greatly successful, they rely on large amounts of labeled training data for their state-of-the-art performance. However, labeled data is not only difficult to obtain for many applications, especially for tasks dealing with sensitive information, but also requires time consuming and costly human annotation efforts. To mitigate label scarcity, recent techniques such as self-training [6, 11] and few shot learning [24, 28] methods have been developed to learn from large amounts of in-domain unlabeled or augmented data. The core idea of self-training is to augment the original labeled dataset with pseudo-labeled data [11] in an iterative teacher-student learning paradigm. Traditional self-training techniques are subject to gradual concept drift and error propagation [29, 24]. In general, data augmentation techniques aim to generate synthetic data with similar characteristics as the original ones. While data augmentation has been widely used

⋆ This research work was conducted while the first author was a Ph.D. student at the University of California, Riverside (sabda005@ucr.edu)

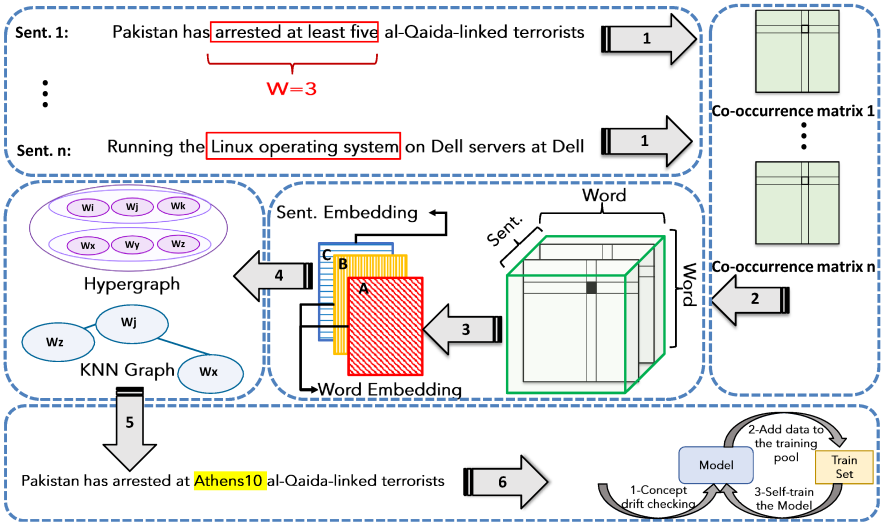


Fig. 1: Overview of the proposed approach.

for image classification tasks [18] leveraging techniques like image perturbation (e.g., cropping, flipping) and adding stochastic noise, there has been limited exploration of such techniques for text classifiers. Recent work on data augmentation for text classification like [28] rely on auxiliary resources like an externally trained Neural Machine Translation (NMT) system to generate back-translations[4] for consistency learning.

In contrast to the above works, we solely rely on the available in-domain unlabeled data for augmentation without relying on external resources like an NMT system. To this end, we develop `Vec2Node` that employs tensor embeddings to consider both the global context and local word-level information. In order to do so, we leverage the association of words and their tensor embeddings with a graph-based representation to capture local and global interactions. Additionally, we learn this augmentation and the underlying classification task jointly to bridge the gap between self-training and augmentation techniques that are learned in separate stages in prior works.

Our contributions can be summarized as follows:

- A novel tensor embedding based data augmentation technique for text classification with few labels.
- A dynamic augmentation technique for detecting concept drift learned jointly with the downstream task in a self-training framework.
- Extensive evaluation on benchmark text classification datasets demonstrate the effectiveness of our approach, particular in low-resource settings with limited training labels along with interpretable explanations.

[4] Process of translating a text to another language and translating it back to the original language.

2 Background

In this section, first, we present mathematical background; then we discuss the problem formulation followed by the details of the proposed method.

2.1 Tensor

A data tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ is a multi-way array i.e., an array with three or more than three dimensions where the dimensions are usually referred to as modes [13].

2.2 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a decomposition technique which factorizes a matrix $\boldsymbol{X}$ into the following three matrices [13]:

$$\boldsymbol{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{1}$$

where the columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal and $\mathbf{\Sigma}$ is a diagonal matrix with positive real entries. A matrix can be estimated by a rank-R SVD as a sum of R rank-1 matrices:

$$\boldsymbol{X} \approx \Sigma_{r=1}^{R} \sigma_{\mathbf{r}} \mathbf{u}_r \circ \mathbf{v}_r \tag{2}$$

2.3 Canonical Polyadic (CP) Decomposition

Canonical Polyadic (CP) or PARAFAC is an extension of SVD for higher mode arrays i.e., tensors [10]. CP/PARAFAC factorizes a tensor into a sum of rank-1 tensors. For instance, a 3-mode tensor is decomposed into a sum of outer products of three vectors:

$$\mathcal{X} \approx \Sigma_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{3}$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$ and the outer product is given by [19, 20]:

$$(\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r)(i, j, k) = \mathbf{a}_r(i)\ \mathbf{b}_r(j)\ \mathbf{c}_r(k)\ \forall i, j, k \tag{4}$$

Factor matrices are defined as $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2 \ldots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2 \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2 \ldots \mathbf{c}_R]$ where R is the rank of the decomposition, which is also the number of columns in the factor matrices. PARAFAC optimization problem is formulated as [13]:

$$\min_{A,B,C} = \|\mathcal{X} - \Sigma_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \tag{5}$$

One effective way to optimize the above problem is to use Alternating Least Squares (ALS) which solves for each one of the factor matrices by fixing the others and cycles over all matrices iteratively until convergence[13].

2.4 KNN Tensor Graph

A k-nearest-neighbor (KNN) graph is a model for representing the nodes in a given feature space such that the k most similar nodes are connected with edges, weighted by a similarity measure [9]. In this work, we use a co-occurrence tensor to map words into an embedding space such that each word (represented by a vector) is a node in the embedding space and then we measure the similarity of the nodes using the Euclidean distance between the corresponding vectors.

2.5 Hypergraph

Hypergraphs [7, 31] are an extension of graphs where an edge may connect more than two nodes to indicate higher-order relationships between the nodes. In contrast to a single weighted connection in traditional graphs, an edge in a hypergraph is a subset of nodes that are similar in terms of features or distance.

3 `Vec2Node` Framework

3.1 Problem Formulation

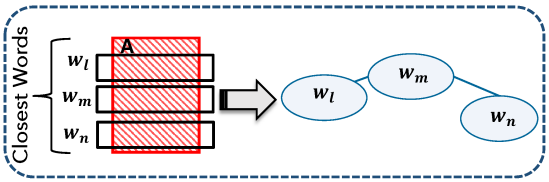
Given a corpus D of labeled data, we aim to **generate** D' that augments D and improves the performance of a classification model M on the downstream task i.e. $f(M(D)) > f(M(D + D'))$, where f is an evaluation measure (e.g., accuracy).

To address the above problem, we propose a novel tensor-based approach for generating synthetic texts from the corpus D. The details of the proposed method, henceforth referred to as `Vec2Node`, are described in the following section.

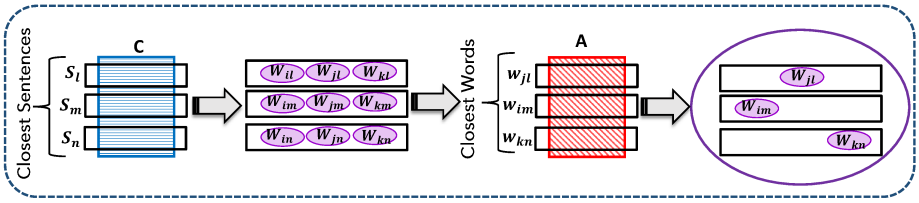
3.2 Data Augmentation

`Vec2Node` leverages tensor decomposition to find word and text embeddings. These are further used for graph-based representations of the word vectors in order to find similar ones as replacement candidates to generate synthetic samples while minimizing the concept drift. `Vec2Node` consists of the following steps:

Tensor-based Corpus Representation Textual content of documents can be represented by a co-occurrence tensor [8, 1] which embeds the patterns shared between different topics or classes. These patterns are formed by words that are more likely to co-occur in documents of the same class. We leverage similar principles to capture existing similarities within a given text. To this end, given a set of samples, we first slide a window of size w across the text of each sample and capture the co-occurring words to represent them in a co-occurrence matrix. Furthermore, we stack the co-occurrence matrices of all samples to form a 3-mode tensor of dimension $T \times T \times S$ where T is the number of terms or words in the entire corpus and S is the number of samples. This process is demonstrated in Figure. 1. The rationale behind this approach is to capture the context (words) for a given target word. In the experimental section, we demonstrate how this approach captures contextually related words.



(a) Graph modeling



(b) Hypergraph modeling

Fig. 2: Graph and hypergraph modeling for representing words' homophily.

Decomposing Tensors into Word and Text Embeddings The objective of this step is to embed the words and the texts of the corpus into rank-R representations which are later used for calculating word similarities. As explained in Section 2, we use CP/PARAFAC to decompose our 3-mode tensor as:

$$\mathcal{X} \approx \Sigma_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{6}$$

Where $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2 \ldots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2 \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2 \ldots \mathbf{c}_R]$ are embedding representations of word, word and text respectively. The word co-occurrence $\mathbf{A}$ and $\mathbf{B}$ are symmetric. Thus, they capture the same information.

Tensor Embeddings for KNN and Hypergraph Homophily Representation In this step, we exploit tensor embedded representations $\mathbf{A}$ and $\mathbf{C}$ to estimate words and texts homophilies (similarities) to find the best candidates for replacement in a given text and generate new synthetic samples. We leverage the following two graph based modelings: **K Nearest Neighbor Graph Modeling**. Consider the factor matrix $\mathbf{A}$ (or $\mathbf{B}$, as they are symmetric and capture the same information) of dimension $N \times R$ where each row is a tensor word embedding in R-dimensional space $\mathbb{R}^R$. We represent the ith row of this matrix which corresponds to word i as a node in R dimensional space. This allows for calculating the Euclidean distance between the nodes and represent the similarity between the nodes (words) as a weighted undirected edge. The Euclidean distance between rows i and j measures the similarity of these two vectors in R-dimensional space. **Hypergraph Modeling**. Spitz et al.[23] propose a hypergraph modeling of the documents where hyperedges are defined by consecutive sentences and words within the text. In that work, the similarity is considered based on spatial closeness. However, in this work, we first leverage the factor matrix $\mathbf{C}$ corresponding to text embedding to find

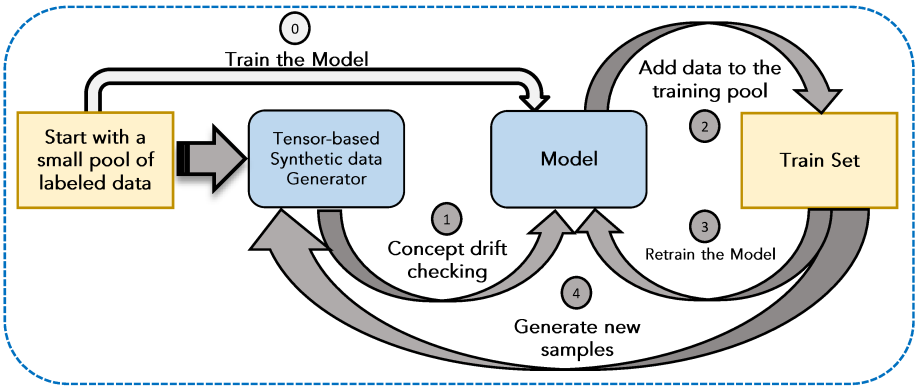


Fig. 3: Few-shot self-training with data augmentation and consistency learning to prevent concept drift.

K closest texts and then we use factor matrix $\mathbf{A}$ to find $K^{'}$ closest words within these K samples. Thus, a hyperedge in this hypergraph is the set of $K^{'}$ closest words. The details of this process are shown in Fig. 2. It is worth mentioning that *our proposed model uses KNN tensor graph* for modeling word similarities. However, for comparison purposes we implement `Vec2Node` framework with hypergraph modeling as well.

3.3 Learning with Data Augmentation and Limited Labels

Contextualized Word Replacement Modeling the corpus using graph or hypergraph representations allows for finding similar words by sorting the edge weights i.e., the Euclidean distances between the nodes, and picking the ones with the smallest distance (i.e., closest words) as the best candidates for replacement and generation of synthetic samples. This process is fully unsupervised given that the tensor decomposition method does not require any labels. Also, it considers local and global contextual information given the graph and tensorial representation of words and texts.

Self-training with Consistency Learning In order to eliminate noisy samples, we check for *concept drift* between the original samples and the synthetic ones using consistency learning in a self-training framework. Given a few labeled samples $\{x_l, y_l\} \in D_l$ for the downstream task, we first fine-tune a base model with parameter θ.

Consider x_u to be the target augmented pair for a source instance x_l generated using the augmentation technique described before. We can use the current parameters θ of the model to predict the pseudo-label for the target x_u as:

$$y_u = argmax_y \; p(y|x_u; \theta) \tag{7}$$

Since the objective of data augmentation is to generate semantically similar instances for the model, we expect the output labels for the source-target augmented pair $\{x_l, x_u\}$ to be similar as well; otherwise, we designate this as a concept drift and discard augmented pairs where $y_l \neq y_u$.

We add the remaining target pseudo-labeled data with consistent model predictions with the source data as our augmented training set $\{x_u, y_u\} \in D_u$ and re-train the base model to update θ. The above steps are repeated with iterative training of the base model with pseudo-labeled augmented data until convergence. The optimization objective for the above self-training process can be formulated as:

$$min_\theta \, \mathbb{E}_{x_l, y_l \in D_l}[-log \; p(y_l|x_l;\theta)] + \lambda \, \mathbb{E}_{x_u \in D_u} \mathbb{E}_{y_u \sim p(y|x_u;\theta^*)}[-log \; p(y_u|x_u;\theta)] \quad (8)$$

where $p(y|x;\theta)$ is the conditional distribution under model parameters θ. θ^* is given by the model parameters from the last iteration and fixed in the current iteration. Similar optimization functions have been used recently in variants of self-training for neural sequence generation [11], data augmentation [28] and knowledge distillation. The details of this process are shown in Figure 3 with the pseudo-code in Algorithm 1.

3.4 Complexity Analysis

In the proposed `Vec2Node` pipeline, the main computation bottleneck is CP decomposition (CPD). In general, CPD is shown to be in the order of the number of non-zero elements [2] of a tensor. In fact, CPD is very fast and efficient for sparse tensors which is the case in this work due to sparsity of the word co-occurrences. Meanwhile, some methods have been proposed for CPD which are amenable to hundreds of concurrent threads while maintaining load balance and low synchronization costs [21]. Moreover, CPD is an offline step in the `Vec2Node` framework i.e., we only execute it once to obtain the embeddings and there is no need to repeat it while training the model.

4 Experimental Evaluation

In this section, we assess performance of `Vec2Node` against baselines we further introduce and then we conduct an ablation study to evaluate components of `Vec2Node`.

4.1 Baselines

- **Base classifiers to asses the effectiveness of augmentation** We compare against the following base classifiers:
  - **`FastText-Softmax`** `FastText` is an efficient word embedding which is an extension of Word2Vec. It represents each word as an n-gram of characters. Thus, in contrast to other non-contextualized embeddings such as GloVe and Word2Vec, provides representations for unseen words [4, 12]. Considering this advantage of `FastText` over mentioned embeddings, we choose `FastText` with a softmax layer (`FastText-Softmax`), as one of our base classifiers.

Algorithm 1 Self-train `Vec2Node`

Input : Base model M, small labeled set D_l.
Return : Self-trained M.

1. Slide a window of size w across the text of each sample in D_l, capture co-occurring words to create a co-occurrence matrix for each sample.
2. Stack all co-occurrence matrices to create a 3-mode tensor $\boldsymbol{\mathcal{X}}$ of size $T \times T \times S$.
3. Decompose $\boldsymbol{\mathcal{X}}$ into **A**,**B**,**C**
4. Use **A**,**C** to model the corpus using graph / hypergraph representations.
5. Calculate Euclidean distances between the nodes to find the closest words.
6. Train M using $D_l = \{x_l, y_l\}$. Set $D = D_l$.
7. While not converged
   - For $\{x_l, y_l\} \in D$, generate augmented samples D'_u by replacing closest words.
   - Assign pseudo-label y_u to each sample $x_u \in D'_u$ using Equation 7.
   - If $y_l = y_u$ then $D = D \bigcup \{x_u, y_u\}$.
   - Retrain M using augmented data D using Equation 8.
8. Return model M

- **`BERT`** leverages contextualized representations using deep bidirectional transformers. We experiment with the pre-trained checkpoints of HuggingFace[5] [26].

– **Neural Machine Translation (`NMT`) to assess the effectiveness of `Vec2Node` augmentation** An Encoder-Decoder architecture with recurrent neural networks (RNN) has become an effective and standard approach for Neural Machine Translation (`NMT`), sequence-to-sequence prediction and data augmentation. `NMT` is the core of the Google translation service [27]. We use `NMT` to translate original sentences into French and then translate them back into English. This process results in synthetic sentences which will be added to the original dataset.
– **`GPT-3` to assess the effectiveness of `Vec2Node` augmentation** Generative Pre-trained Transformer 3 (GPT-3) is an autoregressive language model that generates human-like text. In this work, for each training sample, we generate multiple sentences using `GPT-3` and train a base classifier on the training set, leveraging classic self-training to assign pseudo labels to the generated samples.
– **NLP Word embeddings to assess the effectiveness of tensor embedding** We experiment with the following word embeddings to investigate the efficacy of the tensor embedding in our proposed `Vec2Node` framework. For a fair comparison, for all of the following baselines, we retain KNN graph, self-training and concept drift check components of the proposed `Vec2Node` and only substitute tensor embedding with the following :
  - **`FastText` embedding.** Not only do we use `FastText` for classification, but also we replace the tensor embedding by `FastText` embedding to find the most similar words. we retain other components as mentioned above.
  - **`Word2Vec` embedding.** A shallow 2-layers neural network proposed in [14]. We use `Word2Vec` instead of tensor embedding to find the most similar words using cosine similarity. Similarly, we retain other components in `Vec2Node`.

[5] https://github.com/huggingface/transformers

Dataset	Class	Train	Test	Avg. Words/Doc
SST2	2	67340	872	17
IMDB	2	25000	25000	235
AG News	4	12000	7600	40
DBpedia	14	560000	70000	51

Table 1: Dataset statistics.

  - **Random replacement.** We replace tensor-based similarity strategy by random word replacement while retaining self-training and consistency learning.
- **Matrix modeling (`tf-idf`) to compare the effectiveness of tensor modeling against matrix modeling**: First, we create a `tf-idf` matrix and decompose it into word embeddings using SVD decomposition. Similar to the previous setup, we retain other components in `Vec2Node` and only replace tensor embedding by `tf-idf` embedding. Both random replacement and `tf-idf`, with strong data augmentation and self-training techniques have been shown to obtain very competitive results for text classification [25, 28].
- **Hypergraph similarity representation to assess the effectiveness of KNN graph modeling** We investigate the efficacy of KNN graph modeling against hypergraph modeling proposed in [23]. Similar to the above setup, we only replace KNN tensor graph by hypergraph while keeping other components of `Vec2Node`.
- **`Vec2Node` with and without self-training and consistency learning** We remove the self-training and consistency learning from the `Vec2Node` pipeline to assess the effectiveness of aforementioned mechanisms.

4.2 Evaluation

We experiment on SST2 [22], IMDB [16], AG News [30] and DBpedia [3] with statistics in Table 1, to assess the efficacy of `Vec2Node` on short, long and multi-label datasets respectively. We report results on the corresponding test splits as available from the mentioned works. To facilitate easy comparison, we report relative accuracy improvement (↑) for all the methods over the base model without augmentation.

Base classifiers From Table 2, we observe that `Vec2Node` with tensor data augmentation obtains on average 16.75% and 10.5% improvement over `FastText-Softmax` with no augmentation, using only 1% and 5% of labeled training data respectively. In this experiment, `Vec2Node` is built on top of `FastText-Softmax` to demonstrate the strength of augmentation. We also observe the relative improvement with augmentation to significantly increase with longer text. For example, the improvement in accuracy for IMDB is 16% more than that on SST2 dataset using 5% of labels. This could be attributed to the shorter context samples not being able to generate diverse variety of synthetic samples that are significantly different from the original ones. However, we still demonstrate significant accuracy improvement with augmentation on SST2 as well. In case of DBpedia classification, which is relatively a hard task, `Vec2Node` improves the accuracy of base `FastText-Softmax` by $3-4\%$ using only $1-5\%$ of

Dataset	%Train	#Train	w/o Vec2Node	w/ Vec2Node	Average ↑
SST2	1	673	0.509±0.000	**0.638±0.0007**	**(5.46↑)**
	5	3367	0.710±0.100	**0.740±0.004**	
	100	67340	0.818±0.0018	**0.823±0.0006**	
IMDB	1	250	0.499±0.000	**0.605±0.004**	**(10.26↑)**
	5	1250	0.522±0.012	**0.718±0.001**	
	100	25000	0.857±0.0007	**0.863±0.002**	
AG News	1	1200	0.295±0.003	**0.687±0.023**	**(18.56 ↑)**
	5	6000	0.663±0.001	**0.825±0.002**	
	100	12000	0.900±0.0003	**0.903±0.0008**	
DBpedia	1	5600	0.566±0.000	**0.603±0.000**	**(3.06 ↑)**
	5	28000	0.589±0.015	**0.619±0.000**	
	100	56000	0.602±0.013	**0.627±0.000**	

Table 2: Performance of `FastText-Softmax` classifier with and without `Vec2Node` augmentation.

Dataset	%Train	#Train	w/o Vec2Node	w/ Vec2Node
SST2	0.5	336	0.754	**0.826(7.2↑)**
IMDB	0.5	125	0.776	**0.783(0.7↑)**
AG News	0.5	600	0.869	**0.880(1.1↑)**

Table 3: Performance of `BERT` with and without `Vec2Node` augmentation.

training labels. As illustrated, when we use 100% of the training data, we still observe improvement in classification accuracy which demonstrates the effectiveness of tensor augmentation in both low and high-resource settings.

In contrast to `FastText-Softmax` which is trained from the scratch, the `BERT` model we use here is pre-trained over massive amounts of unlabeled data thereby, works well even in the low-data regime. Thus, to demonstrate the strength of our tensor augmentation i.e., `Vec2Node`, we choose the few-shot setting with only 0.5% of labeled training data. From Table 3, we observe that `Vec2Node` using `BERT` as an encoder along with tensor augmentation to obtain 3% improvement on average over the base `BERT` model using very few training labels. Meanwhile, augmenting SST2, using `BERT` as a classifier, improves the overall performance of `Vec2Node`, where we observe 7.2% improvement of accuracy after augmentation. In case of DBpedia, since it is a very large dataset, even with 0.5% of the labels a pretrained `BERT` achieves its maximum accuracy. Thus we skip it for this experiment. It is worth emphasizing that the pre-trained `BERT` outperforms `FastText-Softmax` which is trained from the scratch. However, in both base model settings, `Vec2Node` improves the performance.

Neural Machine Translation (NMT): As reported in Table 4, `Vec2Node` outperforms `NMT` augmentation strategy as well. We observed that in contrast to synthetic

Dataset	%Train	#Train	w/o Aug.	w/ NMT	w/ GPT3	w/ Vec2Node
SST2	5	3367	0.710±0.100	0.715±0.008(0.50↑)	0.700±0.005(0.01↓)	**0.740±0.004(3.00↑)**
IMDB	5	1250	0.522±0.012	0.692±0.016(17.00↑)	**0.795±0.001(27.3↑)**	0.718±0.001(19.06↑)
AG News	5	6000	0.663±0.001	0.786±0.021(12.30 ↑)	0.801±0.001(13.8↑)	**0.825±0.002(16.20 ↑)**
DBpedia	5	28000	0.589±0.015	0.610±0.005(2.10 ↑)	**0.667±0.060**(7.8↑)	0.619±0.000(3.00 ↑)

Table 4: Performance of FastText-Softmax classifier with augmentations from NMT, GPT-3 and Vec2Node.

samples of Vec2Node, the majority of the synthetic samples created by NMT are quite identical with the original ones and as a result, they do not add diversity to the datasets.

GPT-3 Text Generation: Table 4 also illustrates performance of Vec2Node against GPT-3 on FastText-Softmax classifier. while GPT-3 outperforms Vec2Node by only 1.53% on average (all four datasets), it is also significantly larger with 175 billion parameters compared to Vec2Node with only few hyper-parameters (i.e., R, w and K) as well as pre-trained over massive amount of web corpora.

Ablation Study In this part, we conduct an ablation study to evaluate different components of Vec2Node namely, tensor embedding, KNN tensor graph, and self-training mechanism for few label classification.

NLP Word Embeddings vs. Tensor Embeddings Table 5 demonstrates performance of Vec2Node with different replacement strategies including FastText and Word2Vec. As illustrated, with longer texts as in IMDB and AG News, Vec2Node with tensor embedding, outperforms other word embeddings due to more tangible word co-occurrences in the texts. In case of SST2, where samples are short phrases with fewer co-occurring non-stop words, we observe less diverse synthetic samples. However, we may conclude that tensor embedding outperform other embeddings in general.

Tensor Modeling vs. Matrix Modeling and tf-idf Embedding In addition, Table 5 illustrates the performance of Vec2Node against Random and tf-idf word replacement strategies. Random and tf-idf do not consider the local and global contextual information of the target word during replacement, and, consequently, generate noisy samples. Vec2Node captures both local and global context to outperform these strategies. In case of large datasets such as DBpedia, we observe that matrix modeling results in a very large and memory inefficient representation and suffers from compute bottleneck for SVD decomposition, whereas tensor modeling is memory efficient due to the fact that it breaks down a large co-occurrence matrix into multiple, yet smaller ones.

KNN Graph vs. Hypergraph for Word Similarities From Table 6, we observe that Vec2Node with KNN graph representation to capture word similarities, outperform hypergraph representation on all four datasets. The KNN graph captures globally similar words, whether or not they co-occur in similar sentences, whereas the hypergraph

Dataset	%Train	Matrix (tf-idf)	Random	Word2Vec	FastText	Tensor (Our)
SST2	5	0.733±0.004 (2.3↑)	0.737±0.001(2.7↑)	**0.759±0.03(4.9↑)**	0.730±0.025(2↑)	0.740±0.004(3.0↑)
IMDB	5	0.602±0.021(7.9↑)	0.659±0.013(13.7↑)	0.663±0.01(14.1↑)	0.680±0.045(15.8↑)	**0.718±0.001(19.6↑)**
AG News	5	0.807±0.002(14.3↑)	0.799±0.002(13.6↑)	0.806±0.042(14.3↑)	0.810±0.054(14.7↑)	**0.825±0.001(16.2↑)**
DBpedia	5	Out of Memory	0.619±0.000(3.0↑)	0.619±0.000(3.0↑)	0.619±0.000(3.0↑)	0.619±0.000(3.0↑)
Average↑		6.125↑	8.25↑	9.07↑	8.87↑	**10.45↑**

Table 5: Vec2Node with different word strategies on FastText-Softmax classifier

Dataset	%Train	FastText	Hypergraph	KNN
SST2	5	0.710±0.100	0.722±0.003(1.2↑)	**0.740±0.004(3.0↑)**
IMDB	5	0.522±0.012	0.664±0.004(14.2↑)	**0.718±0.001(19.6↑)**
AG News	5	0.663±0.001	0.811±0.002(14.8↑)	**0.825±0.001(16.2↑)**
DBpedia	5	0.589±0.015	0.615±0.000(2.6↑)	**0.619±0.000(3.0↑)**

Table 6: Vec2Node with KNN vs. hypergraph on FastText-Softmax classifier.

Dataset	%Train	FastText	w/o ST & CL	w/ ST & CL
SST2	5	0.710±0.100	0.720±0.006(1.0↑)	**0.740±0.006(3.0↑)**
IMDB	5	0.522±0.012	0.686±0.005(16.4↑)	**0.718±0.001(19.6↑)**
AG News	5	0.663±0.001	0.791±0.001(12.8↑)	**0.825±0.001(16.2↑)**
DBpedia	5	0.589±0.015	0.614±0.000(2.5↑)	**0.619±0.000(3.0↑)**

Table 7: Vec2Node with and without self-training & consistency learning (ST & CL) on FastText-Softmax classifier.

representation confines the similarity search to words that co-occur in similar texts. This may lead to situations in which all words in a given sentence are replaced by the same word due to lack of candidates in the pool. Moreover, similar words may occur in different contexts and in such cases hypergraph does not capture them.

Vec2Node with and without Self-training and Consistency Learning In this experiment, we ablate the self-training and consistency learning components in Vec2Node to analyze their contribution to the results in Table 7. We observe the self-training component where the model leverages augmented data and pseudo-labels for consistency learning to further improve the performance of Vec2Node by 8.2% on all datasets. Also, this component along with augmentation jointly contributes to 10.45% improvement of Vec2Node over that of FastText-Softmax.

4.3 Interpretability and Examples

Table 8 in Appendix 7, demonstrates synthetic examples from the AG news and SST2 datasets, generated by `Vec2Node` using different word replacement strategies i.e., random, `tf-idf` and tensor embedding. We observe `Vec2Node` to generate better samples with the following features.

Preserving context for word replacement. In contrast to random selection which blindly substitutes words, the co-occurrence based structure of the tensor embedding preserves the context, and selects candidate words that are contextually similar to the original ones. For instance, in example #1 the entity “Jermain Defoe" is replaced by “Owen Michael" as they are more likely to co-occur in a Sport text related to “Real Madrid". As illustrated, the other approaches replace words quite randomly. This feature helps to minimize the concept drift that might happen in the replacement process.

Paraphrasing context. `Vec2Node` leverages a sliding window to capture co-occurring concepts in a sentence, such that non-adjacent words that occur within the same context can be substituted with each other. This contributes to paraphrased sentences generated during augmentation as illustrated in example #2 with re-ordered proper nouns “Samsung" and “SCH-S250".

Tensor embedding preserves word-level similarities. Tensor embedding not only preserves the context-level similarity, but also retains the semantics of the replaced concept. More precisely, it is more likely that a number gets replaced by another number (# 3) or an adverb by another adverb (# 5), and so on and so forth. We observe that not only numbers and verbs, but also prepositions like “a”, “an”, and “the” are replaced by similar concepts in the synthetic samples while preserving the context.

4.4 Related work

Self-training and Consistency Learning Self-training is one of the well-known semi-supervised approaches which has been widely used to minimize the need for annotation leveraging large-scale unlabeled data [15, 11, 17, 24]. For instance, Wang et al. leverage self-training and meta-learning for few-shot training of neural sequence taggers [24]. Moreover, a recent work, a.k.a UDA [28] exploits consistency learning with paraphrasing and back-translation from Neural Machine Translation systems for few-shot learning. In this work, we do not use any external resources such as an NMT system. In fact, we aim to bridge the gap between self-training and augmentation techniques, while solely relying on in-domain unlabeled data for tensor-based augmentation.

5 Conclusion

In this work, we propose a novel tensor-based technique i.e., `Vec2Node`, to augment textual datasets leveraging local and global information in corpus. `Vec2Node` leverages tensor data augmentation with self-training and consistency learning for text classification with few labels. Our experiments demonstrate that synthetic data generated by `Vec2Node` are interpretable and improve the classification accuracy over different datasets significantly in low-resource settings. For instance, `Vec2Node` improves the

accuracy of `FastText` by 16.75% while using only 1% of labeled data. Overall, we demonstrate `Vec2Node` to work well both in low and high-data regime with improved performance when built on top of different encoders (e.g., `FastText`, `BERT`).

6 Acknowledgments

The GPUs used for this research were donated by the NVIDIA Corp. Research was partly supported by a UCR Regents Faculty Fellowship. Research was also supported by the National Science Foundation grant no. 1901379, CAREER grant no. IIS 2046086 and grant no. 2127309 to the Computing Research Associate for the CIFellows project.

References

1. Abdali, S., Shah, N., Papalexakis, E.E.: Hijod: Semi-supervised multi-aspect detection of misinformation using hierarchical joint decomposition. In: ECML/PKDD (2020)
2. Bader, B., Kolda, T.: Algorithm 862: Matlab tensor classes for fast algorithm prototyping. ACM Trans. Math. Softw. **32**, 635–653 (01 2006)
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data **7**(3), 154–165 (sep 2009). https://doi.org/10.1016/j.websem.2009.07.002
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 NAACL. pp. 4171–4186. ACL, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423
6. Du, J., Grave, E., Gunel, B., Chaudhary, V., Celebi, O., Auli, M., Stoyanov, V., Conneau, A.: Self-training improves pre-training for natural language understanding (2020)
7. Gallo, G., Longo, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete Applied Mathematics **42**, 177–201 (04 1993). https://doi.org/10.1016/0166-218X(93)90045-P
8. Guacho, G.B., Abdali, S., Shah, N., Papalexakis, E.E.: Semi-supervised content-based detection of misinformation via tensor embeddings pp. 322–325 (Aug 2018). https://doi.org/10.1109/ASONAM.2018.8508241
9. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. (2011)
10. Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics **16**(1), 84 (1970)
11. He, J., Gu, J., Shen, J., Ranzato, M.: Revisiting self-training for neural sequence generation (2020)
12. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification (2016)
13. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (September 2009). https://doi.org/10.1137/07070111X
14. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. ICML 2014 **4** (05 2014)
15. Li, X., Sun, Q., Liu, Y., Zheng, S., Zhou, Q., Chua, T.S., Schiele, B.: Learning to self-train for semi-supervised few-shot classification (2019)

16. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 142–150. ACL, Portland, Oregon, USA (Jun 2011)
17. Meng, Y., Shen, J., Zhang, C., Han, J.: Weakly-supervised neural text classification. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. ACM (oct 2018). https://doi.org/10.1145/3269206.3271737
18. P. Liu, X. Wang, C.X., Meng, W.: A survey of text data augmentation (2020)
19. Papalexakis, E.E., Faloutsos, C., Sidiropoulos, N.D.: Tensors for data mining and data fusion: Models, applications, and scalable algorithms. ACM Trans. Intell. Syst. Technol. **8**(2), 16:1–16:44 (Oct 2016). https://doi.org/10.1145/2915921
20. Sidiropoulos, N., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E., Faloutsos, C.: Tensor decomposition for signal processing and machine learning. IEEE Transactions on Signal Processing **PP** (07 2016). https://doi.org/10.1109/TSP.2017.2690524
21. Smith, S., Ravindran, N., Sidiropoulos, N.D., Karypis, G.: Splatt: Efficient and parallel sparse tensor-matrix multiplication. In: IPDPS. pp. 61–70 (2015)
22. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1631–1642. ACL, Seattle, Washington, USA (Oct 2013)
23. Spitz, A., Aumiller, D., Soproni, B., Gertz, M.: A versatile hypergraph model for document collections. SSDBM 2020 (2020)
24. Wang, Y., Mukherjee, S., Chu, H., Tu, Y., Wu, M., Gao, J., Awadallah, A.H.: Adaptive self-training for few-shot neural sequence labeling. ArXiv **abs/2010.03680** (2020)
25. Wei, J., Zou, K.: EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In: EMNLP-IJCNLP. pp. 6383–6389. Association for Computational Linguistics, Hong Kong, China (Nov 2019)
26. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 38–45. ACL, Online (Oct 2020)
27. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J.R., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G.S., Hughes, M., Dean, J.: Google's neural machine translation system: Bridging the gap between human and machine translation. ArXiv **abs/1609.08144** (2016)
28. Xie, Q., Dai, Z., Hovy, E.H., Luong, M., Le, Q.V.: Unsupervised data augmentation. CoRR **abs/1904.12848** (2019)
29. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. ICLR (2017)
30. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28, pp. 649–657. Curran Associates, Inc. (2015)
31. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. vol. 19, pp. 1601–1608 (01 2006)

7 Appendix

Hyper-parameter configurations. We perform grid search for the window size $w \in [3-10]$, $R \in [5-150]$, with the best value as $w = 3$ and $w = 7$ for short and long texts respectively. Similarly, we found the best value for the decomposition rank $R = 25$. We also perform a search for the word replacement ratio – where we replace words by their nearest neighbors in the embedding space to generate augmented samples (as outlined in Section 2). We observe that replacing $20-25$ % of the words results in fewer number of concept drift which consequently leads to the best accuracy. We also explored the possibility of consecutive vs. random word replacement. We found the classification accuracy of the former to be 3% higher as it preserves the context resulting in $10-12$ % less concept drift. Reported experimental results are averaged over 25 runs with different random seeds. For `BERT` experiments, we use self-trianing algorithm 1, and for `FastText` and all other baselines we use classic self-training due to the speed. The code is publicly available for reproducibility purposes[6].

#	Original-Label	Augmented-Method	Closest Tensor Neighbors
1	Jermain Defoe may replace him for England on Saturday-**Sports**	Owen Michael may replace him for England on Saturday-**Tensor** Jermain Defoe may replace him for England vast desert Saturday-**Random** Jermain Defoe may replace 1.22 for England 0-11 Saturday 1,070-**Tf-idf**	Jermain ↪ Owen Defoe ↪ Michael
2	The Samsung SCH-S250 5-megapixel camera phone will enable users to take photos-**Sci/Tech**	SCH-S250 Samsung the 5-megapixel camera phone will enable users to take photos-**Tensor** seem dominance in hurricane season SCH-S250 Samsung The will enable users to take photos -**Random** barbarians SCH-S250 Samsung The will enable users to take photos-**Tf-idf**	Samsung ↪ SCH-S250 SCH-S250 ↪ Samsung
3	Pakistan has arrested at least five al-Qaida-linked-**World**	Pakistan has arrested at athens 10 al-Qaida-linked -**Tensor** Pakistan has arrested .26 pairings fatter al-Qaida-linked-**Random** Pakistan has 0.84 , 1,000-yard 1,000-yard al-Qaida-linked-**Tf-idf**	least ↪ athens five ↪ 10
4	working from a surprisingly sensitive script co-written by gianni romoli...**Positive**	add a surprisingly sensitive script co-written by gianni romoli... **Tensor** working from a surprisingly sensitive script co-written by gianni mixed second-rate -**Random** working from a surprisingly sensitive script co-written by becalmed chillingly ...-**TF-idf**	working ↪ add from ↪ ' '
5	Never seems hopelessly juvenile .-**Negative**	Never seems amateurishly accused-**Tesnor** Never seems dawn carmichael-**Random** Never seems born actioners -**Tf-idf**	hopelessly ↪ amateurishly juvenile ↪ accused

Table 8: Snapshot of similar contexts captured with tensor embedding and augmented sentences created by replacing 20-30% words in a sentence.

[6] https://github.com/Saraabdali/Vec2Node