Investigating semi-supervised methods for enhancing text transformers with dimension reduction techniques

Yu Hei Chau
20644747
HKUST
yhchau@connect.ust.hk

Abstract

Transformers have been highly successful in producing word embeddings, which are numerical representations of words that can be used in natural language processing tasks such as language translation, sentiment analysis, and text classification. However, it usually requires a lot of resources to train a transformer with a large context length. This project seeks to investigate whether dimension reduction techniques can be applied to enhance the performance of transformers under a resource constrained environment.

1 Background

Transformers such as BERT, sBERT [2] have revolutionized the field of natural language processing by producing highly effective word embeddings. These embeddings capture the complex relationships between words and can be used to improve the performance of a wide range of language-related tasks, including language translation, text classification, and sentiment analysis. The success of transformers can be attributed to their ability to leverage self-attention and contextual information, allowing them to produce highly accurate embeddings that can capture the nuances of language.

However, training or fine-tuning transformers requires a significant amount of GPU memory, which can be a challenge in resource-constrained environments. Due to the large number of parameters and the complex architecture of transformers, training them requires significant computational resources, particularly when dealing with large datasets or complex models. Fine-tuning a mpnet sentence-transformer requires 40GB of VRAM for a moderate batch size, which is not available on many consumer devices.

This can make it difficult for researchers or organizations with limited resources to fully leverage the power of transformers. It is also hard for localized personalized algorithms that uses text-classification or sentiment learning to be implemented on popular mobile devices, where data is preferred not to be uploaded to a cloud computer server due to privacy concerns. As a result, there is a growing need for more efficient algorithms and hardware to make transformer-based models more accessible to a wider range of users and use cases.

1.1 Dataset and problem

A general formulation of the problem is to perform text-classification or sentiment learning for a dataset, where each data contains a large number of sentences, not necessarily in sequential order. There may be extra structure (such as graph) in the hypothetical dataset. For this project, we focus our attention on the Kaggle Learning Equality Competition [1].

The competition poses an information retrieval problem, in the context of assigning learning materials to school topics. More concretely, we are given two sets T, C where T are the set of all topics and C

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

are the set of all contents. The goal is to predict Corr(t, c), meaning that the topic correlates with the content. For example, Shakespeare's Macbeth should correlate to an English Literature learning topic, but not a Mathematics topic.

Each topic $t \in T$ and content $c \in C$ contains textual information describing the learning topic and learning materials respectively. The topics have an extra tree structure between them, where subtopics could be classified under a broader topic. For example, the topics Algebra and Calculus would be under Mathematics, and Integration and Differentiation would be under Calculus. Formally, the set of all topics T forms a partition, with each connected component $T_{\textit{channel}}$ being a acyclic graph with exactly one root element $t_{root,channel} \in T_{\textit{channel}}$. The tree structure can be visualized via tree_structure_visualization.py.

The textual information are description(t), description(c), title(t), title(c) and text(c). The text content is not provided for topics t, we only have the description and title. The are additional information such as language and copyright holder for both of the fields, but will be omitted in the subsequent analysis.

1.2 Preprocessing

The textual information comes in different languages, and may have tokens unrelated to its semantic content (such as HTML code) as they are probably scraped on the web by Kaggle staff. To convert each topic $t \in T$ and content $c \in C$ into a vector representation, some preprocessing is done to hopefully remove the unwanted tokens, combine text(c) and description(c) into description(c) for contents $c \in C$, and then use OpusMT pretrained language translate models along with ctranslate2 package to convert the textual information to English. Finally, we use a pretrained transformer by sentence-transformers to obtain a 768 dimensional vector for each $t \in T$ and $c \in C$.

1.3 Formulation of the semi-supervised learning problem

In this project, we assume that expensive computational devices are unavailable, or the context we develop the algorithm for is on a resource constrained device. Therefore we would only make use of transformers for inference, but we would not train or fine tune transformers. The goal is to investigate whether a fixed pretrained transformer model can be enhanced with some semi-supervised learning techniques.

Recall that there are T, C and a map Corr: $T \times C \to \{0, 1\}$. Here, we assume that Corr$|_{T' \times C'}$ is known where $T' \subset T, C' \subset C$ are strict subsets, and from this information, the goal is to extrapolate and predict the whole map Corr: $T \times C \to \{0, 1\}$. This can be viewed as a semi-supervised learning problem, where we only have labels on a subset $T' \times C'$, but we also make use of the entire distribution $T \times C$ for learning.

In the project, we pick $T = T_1 \sqcup T_2, C = C_1 \sqcup C_2$, and we only make use of the information of Corr on $T' = T_1, C' = C_1$. The subsets are picked so that Corr$(t_i, c_j) = 0$ whenever $t_i \in T_i, c_j \in C_j$ and $i \neq j$, and every two elements $t, t' \in T$ belonging to the same channel (graph / tree) must either both belong to T_1 or both belong to T_2. In other words, the train/test sets are picked so that no correlation occurs between the train and test sets, and each topic tree must entirely belong to either the train set or test set. In practice, $|T_2| \geq \frac{1}{2}|T_1|$ and $|C_2| \geq \frac{1}{2}|C_1|$.

Supervised approach For the classifier, we use a standard approach of using a deep neural network $f_\theta : \mathbb{R}^{1536} = \mathbb{R}^{768} \times \mathbb{R}^{768} \to [0, 1]$ to approximate the map Corr: $T \times C \to \{0, 1\}$. Recall that in Section 1.2 that T, C are represented by $\mathbb{R}^{768}$ using sentence-transformer text embeddings. To be faithful to the resource limited constraints, the network is represented by a fully connected neural network with 5 layers, and not any larger. We train the network with the standard tricks of using binary cross entropy loss, Adam optimizer and so on. This is clearly a supervised approach, and can be viewed as training a transformer network but with the weights of the transformer fixed, since the word embeddings are obtained through a transformer.

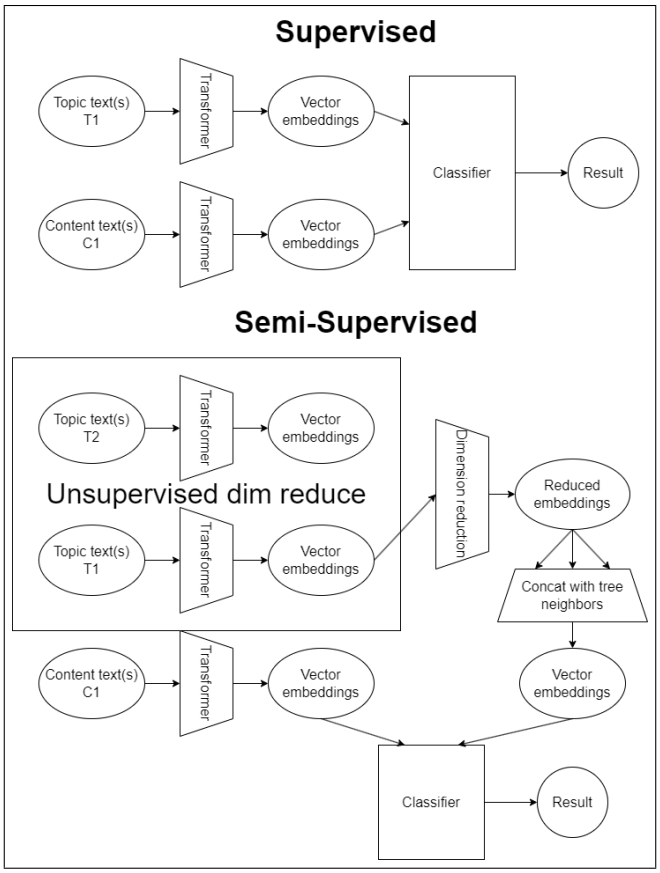


Figure 1: Supervised and semi-supervised approaches

Semi-supervised approach Many textual analysis problems have extra structure for the data, such as the tree structure for the topics in our problem. However, the previous approach does not make use of the tree structure. We expect that information about the topic node's neighbors be useful for textual analysis. For example, knowing that Trigonometry is related to Mathematics might be useful for the transformer to pick up on the 'Mathematics' keyword, where the transformer is not specifically fine-tuned to identify the keyword Trigonometry.

A naive approach would be to concatenate the text information description(t) with its neighbors description(t') for $t' \in \textit{neighbors}(t)$, and then plug it directly into the transformer. However, the text would be quite long and transformers with large context length are costly to fine tune, while a pretrained transformer not specifically designed for this task may not opt to identify the subtle differences inside a long text.

Another approach is to append the 768-dimensional embedding vectors together with its neighbors, such that

$$\textbf{Vect}_{new}(t) = \textit{Concat}_{t' \in \textit{Neighbors}(t)} \textbf{Vect}(t')$$

However, this makes the input for f_θ very high dimensional, which may require a larger network to be able to optimally learn the patterns of Corr: $T \times C \to \{0, 1\}$, which defeats the purpose of investigating on resource constrained situations.

Therefore, the proposal is to first do unsupervised dimension reduction on the embedding vectors

$$\textbf{Vect}(T) \subset \mathbb{R}^{768}$$

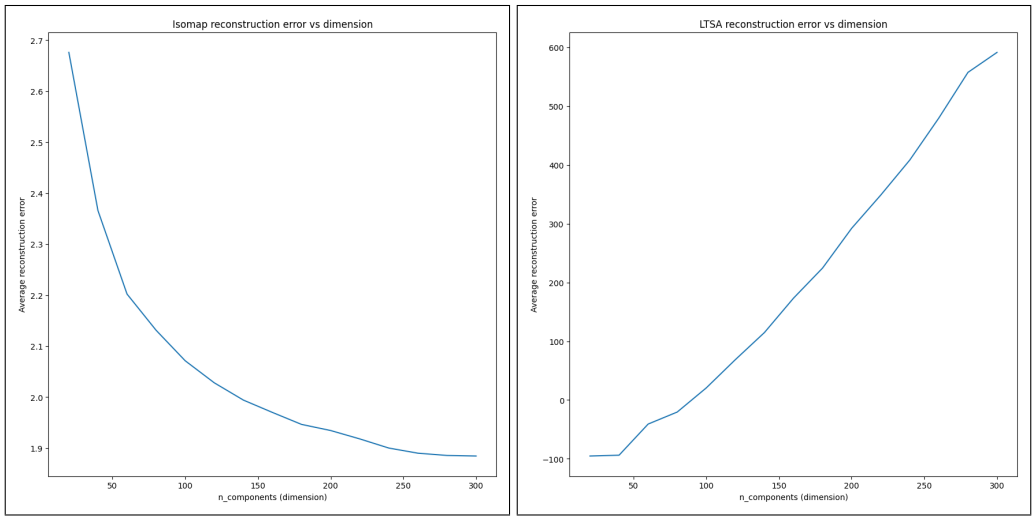
and then use a same concatenation method so that $\textbf{Vect}_{new}(t) \in \mathbb{R}^{768}$ always, but now the vectors contain "information" about its neighbors. Afterwards, we use the network with same architecture and training methods f_θ to learn Corr: $T \times C \to \{0, 1\}$ with the new embeddings.

2 Dimension reduction and tree neighbors

2.1 Dimension reduction

We use PCA, isomap and LTSA to reduce the dimension of the embedding vectors. It is of course possible to also try out other manifold learning methods introduced in the lectures such as SNE, TSNE and so on. In the context of the project, the vectors themselves are obtained using a transformer model, which in itself can be considered as a latent space. It is therefore hard to argue what the embedding vectors submanifold would look like, and to further give theoretical justifications for which manifold learning methods would be better. As a result, we simply pick some common dimension reduction methods and probe their performance.

It is quite straightforward to interpret what PCA is doing. On the other hand, we plot the isomap and LTSA reconstruction errors versus the reduced dimension:



It seems that the reconstruction error stabilizes at $n \geq 300$ reduced dimensions, so the submanifold of learning topics text embedding vectors does not really have 'small dimensions'. Since the LTSA method is quite unstable, it will be omitted in subsequent analysis.

2.2 Tree neighbors

Recall that the $T = T_1 \sqcup T_2$ where T can be partitioned into channels (trees) such that each partition is either entirely contained by T_1 or T_2. We can make use of the tree information to infer textual information about the topic $t \in T$. Any tree $T_{\textit{channel}} \subset T$ can of course be considered as a graph (V, E) with vertices $V \subset T$. We intuitively expect the close neighbors to be "more similar" to t rather than far away ones, for example "Calculus" would be more similar to "Integration" than "Mathematics" to "Integration". Define the distance

$$d(t_1, t_2) = \textbf{Unique shortest path from } t_1 \textbf{ to } t_2 \qquad \forall t_1, t_2 \in T_{\textit{channel}}$$

To obtain a modified 768 dimensional vector, we first fix $m = 4, 8, 16$.

Algorithm 1: Algorithm for computing the final text embeddings with tree neighbors

Parameters : m, similar_or_disimilar

Vect' = Dimension reduction on **Vect** to $\frac{768}{m}$ dimensions over all $t \in T$
Initialize **Vect**$_{new}$
for $t \in T$ **do**
    list = [t]
    $D = 1$
    while $length(list) < m$ **do**
        if $|\{d(t', t) = D\}| \leq m - length(list)$ **then**
            Append to list all t' such that $d(t', t) = D$
        else
            Append to list $m - length(list)$ **most similar or dissimilar** t' such that $d(t', t) = D$
        end
        $D = D + 1$
    end
    Vect$_{new}(t) = concat($**Vect'**$(list))$
end

3 Results of semi-supervised training

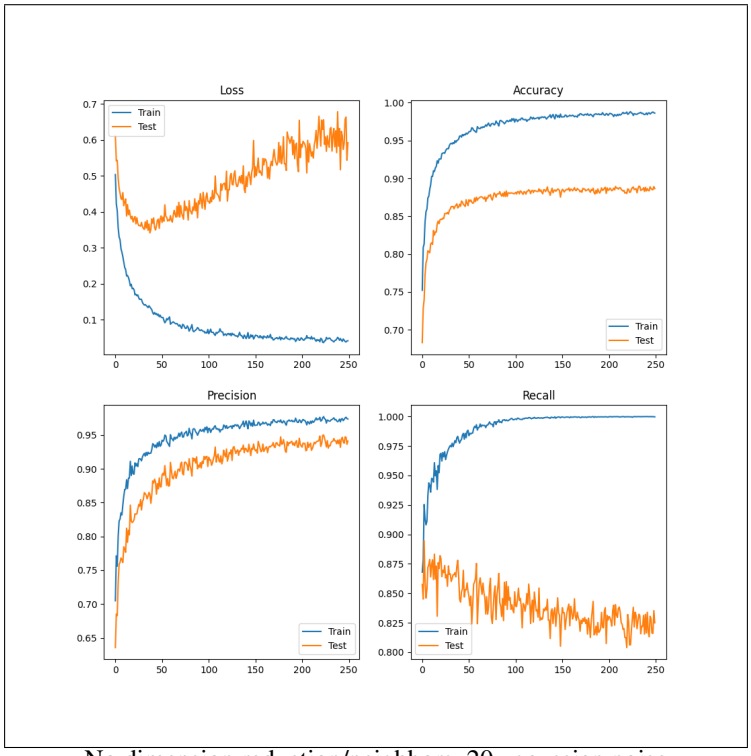
Now we train a network $f_\theta : \mathbb{R}^{768} \times \mathbb{R}^{768} \to [0,1]$ to approximate Corr: $T \times C \to \{0,1\}$. Our goal is for $f_\theta(\mathbf{Vect}_{new}(t), \mathbf{Vect}(c)) \sim \mathit{Corr}(t,c)$. To emulate a semi-supervised approach, we expose only $\text{Corr}|_{T_1 \times C_1}$ in the training and backpropagation, and use $\text{Corr}|_{T_2 \times C_2}$ for evaluation only. Recall that $\text{Corr}|_{T_1 \times C_2} = 0$ and $\text{Corr}|_{T_2 \times C_1} = 0$ (see Section 1.3).

Hyperparameters and training method The network architectures and training methods are the same throughout, except the Gaussian noise for input. It is a standard 5 layered neural network with ELU activations and sigmoid activation in the last neuron, and binary cross entropy is used as the loss function. Since the correlations are quite sparse ($P(\mathit{Corr}(T,C) = 1) \approx 0$ when T, C are considered as random variables), directly training such a classifier will quickly bias the the neural network to classify everything as 0. Therefore we sample using an alternative method, first from $S = \{(t,c) \in T_1 \times C_1 : \mathit{Corr}(t,c) = 1\}$ and from $T_1 \times C_1$ with same size. The same is done for the test set $T_2 \times C_2$. This makes $P(\mathit{Corr}(T,C) = 1) \approx 0.5$ (slightly larger).

For better generalization, we employ standard tricks such as dropout and weight decay to the training procedure, which is kept the same throughout different unsupervised reduction methods. The input Gaussian noise roughly corresponds to how far away from a point $p \in T_1 \times C_1$ the network should generalize to. We expect this might depend on the type of dimensionality reduction, as different dimensionality reduction methods distort the distances in different ways. Therefore we do a parameter search on the noise, apart from the n_neighbors and dimension reduction method.

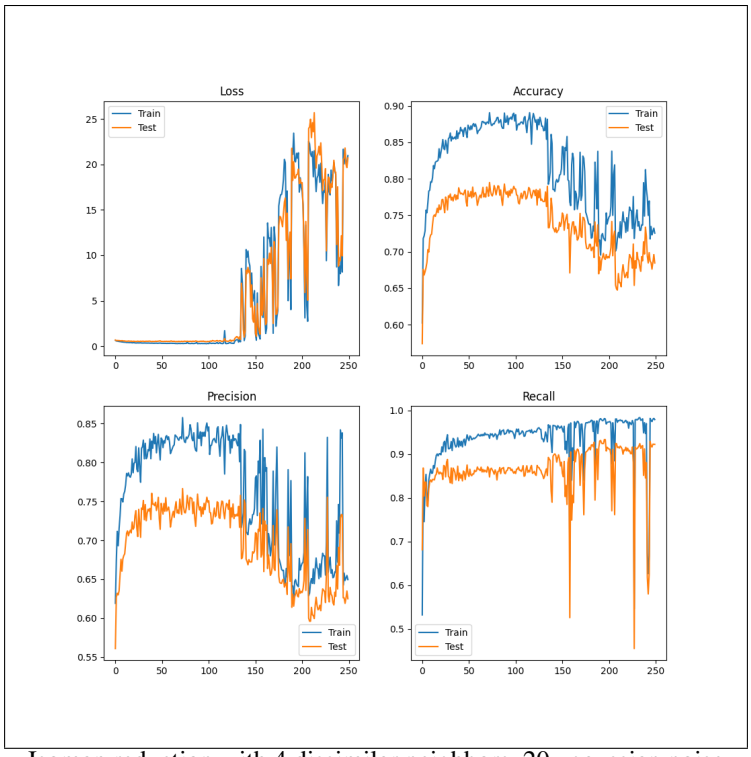
3.1 Training curves

If the Gaussian noise is picked appropriately, the accuracy of the network is stable across the training epochs. However, the accuracy is traded for recall, or vice versa, and the cross entropy loss for the test set ($T_2 \times C_2$) increases. Of course, all the metrics in the training set ($T_1 \times C_1$) improves over epochs. This suggests overfitting causes the network f_θ is reject samples too well or accept samples too well, sacrificing the other, but the overall accuracy is similar.



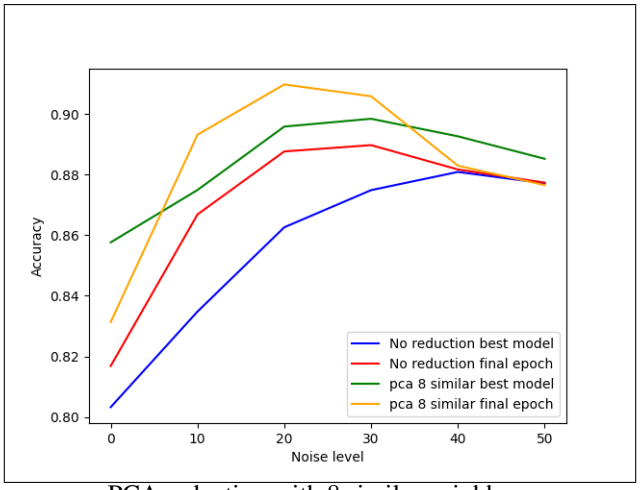
No dimension reduction/neighbors, 20x gaussian noise

If the Gaussian noise is too large, the training of the network would be unstable.

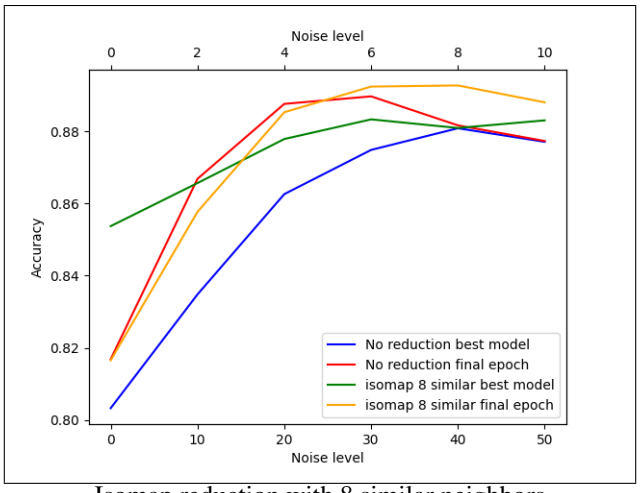


Isomap reduction with 4 dissimilar neighbors, 20x gaussian noise

Empirically, we observe that the noise applied to isomap is best at $0 - 10$, and the noise applied to PCA or no neighbors/dimension reduction is $0 - 50$. The larger the noise, the more the network f_θ favours the recall in the test set. Here is a plot of accuracy versus the input Gaussian noise:



PCA reduction with 8 similar neighbors



Isomap reduction with 8 similar neighbors

4 Conclusions

These conclusions show that the graph structure of the topics provide valuable information that increases the accuracy on the test set in a semi-supervised manner, rather than directly using the pretrained transformer embeddings in a purely supervised manner.

Interestingly, the algorithms do pretty well on PCA and expectedly better than without using any information of neighboring text. Surprisingly, isomap performs worse than PCA, which may be caused by the landmark choice-like algorithm isomap in sklearn, so isomap is approximated rather than computed on the whole set.

References

[1] Learning equality - curriculum recommendations. Kaggle. (n.d.). https://www.kaggle.com/competitions/learning-equality-curriculum-recommendations/overview

[2] Sentence transformers documentation. SentenceTransformers Documentation - Sentence-Transformers documentation. (n.d.). https://www.sbert.net/