Invernet: An Inversion Attack Framework to Infer Fine-Tuning Datasets through Word Embeddings

Ishrak Hayet, Zijun Yao, Bo Luo
University of Kansas
{ishrakhayet, zyao, bluo}@ku.edu

Abstract

Word embedding aims to learn the dense representation of words and has become a regular input preparation in many NLP tasks. Due to the data and computation intensive nature of learning embeddings from scratch, a more affordable way is to borrow the pretrained embedding available in public and fine-tune the embedding through a domain specific downstream dataset. A privacy concern can arise if a malicious owner of the pretrained embedding gets access to the fine-tuned embedding and tries to infer the critical information from the downstream datasets. In this study, we propose a novel embedding inversion framework called *Invernet* that materializes the privacy concern by inferring the context distribution in the downstream dataset, which can lead to key information breach. With extensive experimental studies on two real-world news datasets: Antonio Gulli's News and New York Times, we validate the feasibility of proposed privacy attack and demonstrate the effectiveness of *Invernet* on inferring downstream datasets based on multiple word embedding methods.

1 Introduction

Discriminative representation learning is a critical factor in determining the success of machine learning models. In Natural Language Processing (NLP), many deep neural networks need the input words or documents to be represented in a form of numerical vectors which indicate the semantic distinction among information. Different methods of converting vocabulary tokens to representations have been discussed in literature Wang et al. (2020). For example, an intuitive way of converting categorical vocabulary to numerical vectors is to one-hot encode the tokens into a bitstring or commonly known as a bag of words representation Harris (1954). However, one-hot encoding can easily result in high-dimensional representations and incapability of accommodating unseen information. Another way is using distributed representation called word embedding Mikolov et al. (2013a), which is low-dimensional and has become increasingly popular by capturing complex and subtle semantic difference of input words or documents Babić et al. (2020).

In practice, well trained word embeddings demand large-scale training corpus, sufficient training iterations, and high computational capacity. For example, Strubell et al. (2019) shows that training a BERT-base model emits around 1,500 lbs of carbon dioxide and costs around \$3,000 - \$12,000. Ensuring such environmentally and financially costly training factors are not economical for most small organizations and individuals. As a result, large organizations (e.g., Google Cloud, Hugging Face) with computational resource would train word embeddings on large and generic datasets, and offer the pretrained embeddings to public through platforms like Tensorflow Model Hub or Hugging Face Model Repository Wolf et al. (2020). With the pretrained embedding in hand, public users can further refine the embeddings by training on their own datasets. As a result, domain adapted downstream embeddings can be generated with much smaller corpus and cost. Such refinement process is known as fine-tuning and can be considered as the de facto standard practice of how word representations are commonly learned in today's natural language processing tasks Ruder (2021), such as the biomedical domain Gu et al. (2021) and the financial domain Araci (2019), etc.

Often times, users of pretrained word embeddings think that the embeddings are simply dense vector representations that nothing much can be inferred about the training corpus from a collection of numbers. So, most of the times, they do not exercise caution in keeping their downstream word embeddings secure. For this widespread case, we attempt to address a question: is it safe to release fine-tuned embeddings if the downstream dataset is sensitive? Specifically, by assuming that the fine-tune embeddings inherit certain optimization pattern from pretrained embeddings, the inference of a downstream dataset can be made inversely from the fine-tuning embedding if we know the examples of how the pretrained embeddings are derived at the beginning. Therefore, in this work, we consider a particular attack model: when the malicious party knows the pretrained datasets (e.g., pretrained embedding publishers), and gets the access to fine-tuned embeddings, is it feasible to infer the semantic information of downstream corpora such as word co-occurrence statistics?

Attacks on machine learning models have been studied extensively Pitropakis et al. (2019). In Shokri et al. (2017), the authors train adversarial shadow models to carry out membership inference attacks against different commercial models. Feature-based analyses of information leakage in machine learning models are discussed in Chen et al. (2020), and Song and Raghunathan (2020). In Chen et al. (2020), the authors consider the owner of the downstream dataset as the attacker and perform a feature alignment based attack on the source dataset in a deep transfer learning setting. In Song and Raghunathan (2020), the authors try to infer the constituent words of a sentence given that they have access to sentence embedding. However, our proposed research problem is unique in that we consider the owner of the source dataset to start the attack on the downstream dataset. With the advantage of having inference datasets similarly distributed as source datasets, the source dataset owner can simulate fine-tune training multiple times and study the fine-tuning results based on a particular input. Therefore, the attackers can inversely infer how the downstream dataset looks like by comparing the resulting embedding with pre-trained embedding. Furthermore, the attack strategy makes the inference model-agnostic, thereby rendering our proposed problem and method generic across various embedding models, such as Word2Vec and GloVe.

Specifically, we develop an inversion framework *Invernet* for attacking downstream dataset based on leaked fine-tuning embeddings. The primary components of *Invernet* architecture are a focused document sampling scheme, and a deep-learning-based inference model. The goal of the focused sampler is to ensure that the selected samples for inference will have reduced variance. Then the

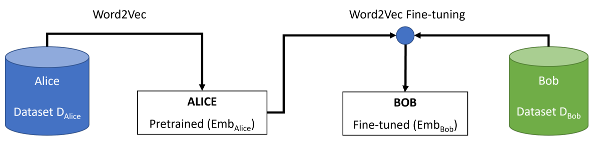

Figure 1: Attack model.

inference model is designed as an ensemble classification problem where multi-layer perception models try to learn the inverse relationship between the fine-tuning embeddings and the context information in the downstream dataset from different inference samples. In the experimental study, We validate the performance of the proposed framework in terms of how much context information of the downstream dataset is predicted, and we illustrate the utility of context prediction through the experiments on membership attack. Based on the results on two real-world datasets and three embedding models, our work validates the privacy concern of publishing fine-tuned embeddings, and demonstrates the effectiveness of proposed inference framework. The contributions of our work are as follows:

- Presenting a novel privacy problem of using fine-tuned embeddings to infer the contextual information of downstream datasets.
- Designing an inference attack framework using neural network with focused inference sampling strategy to accurately predict the context information of downstream datasets.
- Conducting comprehensive experimental study on two real-world datasets including AG_News and New York Times. Results demonstrate the advantage of the proposed Invernet framework over all the baselines.

2 Attack Model and Problem Statement

As shown in Fig. 1, Alice is a *malicious user* (attacker) who has the computational power to train complex embedding models from a large dataset $\mathcal{D}^{\text{Alice}}$ and publishes her pre-trained word embeddings $\mathbf{\Phi}^{\text{Alice}}$. Bob, the victim, downloads and fine-tunes $\mathbf{\Phi}^{\text{Alice}}$ on his smaller dataset $\mathcal{D}^{\text{Bob}}$. $\mathbf{V}^{\text{Alice}}$ and $\mathbf{V}^{\text{Bob}}$ are sets of all the unique words from $\mathcal{D}^{\text{Alice}}$ and $\mathcal{D}^{\text{Bob}}$ respectively, excluding the stop-words. The formal assumptions are as follows:

- Alice owns $\mathcal{D}^{\text{Alice}}$ and $\mathbf{\Phi}^{\text{Alice}}$.
- Alice can compute a binary word-to-word co-occurrence $\mathbf{C}^{\text{Alice}}$ of $\mathcal{D}^{\text{Alice}}$.

Table 1: Binary co-occurrences of the word "economy" in $\mathcal{D}_{\text{Bob}}$ within a window size $\text{W}_\text{C} = 1$ after removing stop words from Sentence 1="Condition of economy is good" and Sentence 2="Economy is growing".

Vocab	condition	economy	good	growing
Sentence 1	1	0	1	0
Sentence 2	0	0	0	1

- Alice gets access to fine-tuned embeddings $\mathbf{\Phi}^{\text{Bob}}$ either from the public domain (e.g. Bob shares it) or by malicious means (e.g. through an attack or an insider).
- $\mathbf{V}^{\text{Bob}} \cap \mathbf{V}^{\text{Alice}} \neq \emptyset$.

Alice's attack objective is to obtain the private semantic information in $\mathcal{D}^{\text{Bob}}$. Given the fine-tuned embeddings from Bob, we formulate the task as predicting the word-to-word co-occurrence statistics among Bob's vocabulary, considering that co-occurrence statistics can truly reflect the semantic information of a dataset. Therefore, the question we are asking is: "*Can Alice learn the binary word-to-word co-occurrence* $\mathbf{C}^{\text{Bob}}$ *from Bob's fine-tuned word embeddings* $\mathbf{\Phi}^{\text{Bob}}$ *with the knowledge of* $\mathcal{D}^{\text{Alice}}$, $\mathbf{C}^{\text{Alice}}$, *and* $\mathbf{\Phi}^{\text{Alice}}$*?*" Formally, we define our question as a learning problem where Alice learns a set of mapping functions $\mathbf{F}$, such that

$$\mathbf{F} = \{ f^{\text{word}} \mid \forall \text{ word} \in \mathbf{V}^{\text{Alice}} , \quad f^{\text{word}} : < \mathbf{\Phi}^{\text{Alice}}, \mathbf{\Phi}^{\text{Bob}} > \rightarrow \mathbf{C}^{\text{Bob}}(\text{word}) \} \tag{1}$$

Table 1 shows an example of the output co-occurrence vectors for the target word "economy".

3 Methodology

3.1 Pretraining

We use different embedding models like Word2Vec Mikolov et al. (2013a),and GloVe Pennington et al. (2014) to pretrain word embeddings. Word2Vec model can be trained using either Continuous Bag-of-words (CBOW) or SkipGram method. CBOW method maximizes the log probability of a word given a set of surrounding words and SkipGram method maximizes the average log probability of the contextual words given a target word. For a window range of [-c, c] around a word w_t at position t, CBOW and SkipGram methods optimize the following Eqs. (2) Mikolov et al. (2013a) and (3) Mikolov et al. (2013b) respectively. For GloVe, we optimize the following Eq. (4) where X_{ij} is the number of times words w_i and w_j co-occur within a certain window, v_i and $\tilde{v_j}$ are the word vectors of w_i and w_j respectively Pennington et al. (2014).

$$\log p(w_t | w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c}) \tag{2}$$

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \tag{3}$$

$$\sum_{i,j} f(X_{ij})(v_i^\top \tilde{v_j} - \log X_{ij})^2 \tag{4}$$

where T is the total number of words in a specific sequence. In Eq. (3), $p(w_{t+j}|w_t)$ is fundamentally defined as a softmax probability Mikolov et al. (2013b) that can be computed using either a hierarchical softmax or negative sampling for computational feasibility Mikolov et al. (2013b).

3.2 Fine-tuning

Fine-tuning is the process of updating pretrained word vectors based on a downstream domain or downstream task or both. When training a model from scratch, the word vectors can be initialized to either constant or random values sampled from a specific distribution. However, during fine-tuning, the word vectors are initialized using pretrained vectors and then updated using gradient descent method as follows:

$$\theta^t = \theta^{t-1} - \eta \nabla_\theta J(\theta^{t-1}) \tag{5}$$

where θ^t is the state of the weight vector at epoch t, η is the learning rate, and ∇_θ is the gradient of the cost function $J(\theta)$.

3.3 Invernet Framework

The Invernet framework consists of two primary components namely a focused inference data sampler and an inference model. The objective of the Invernet framework is to learn - *how embedding of a word changes under the impact of particular context distributions during the fine-tuning process.* Therefore, supplying a large number of diverse fine-tuned word embedding matrices to the inference model will enable us to better learn the fine-tuning changes.

First, as Fig. 2 showing the inference sampler of Invernet, the attacker Alice's source dataset is

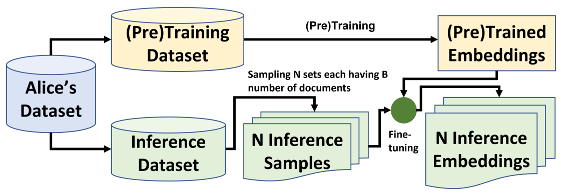

Figure 2: Inference sampler.

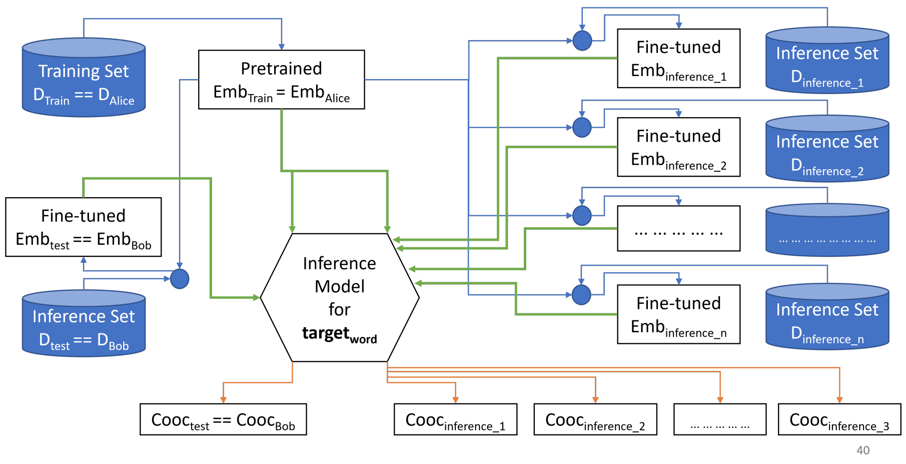
Training Set $D_{Train} == D_{Alice}$
Pretrained $Emb_{Train} = Emb_{Alice}$
Fine-tuned $Emb_{test} == Emb_{Bob}$
Inference Set $D_{test} == D_{Bob}$
Inference Model for $\textbf{target}_{word}$
Fine-tuned $Emb_{inference\_1}$
Inference Set $D_{inference\_1}$
Fine-tuned $Emb_{inference\_2}$
Inference Set $D_{inference\_2}$
… … … … …
… … … … … … … …
Fine-tuned $Emb_{inference\_n}$
Inference Set $D_{inference\_n}$
$Cooc_{test} == Cooc_{Bob}$
$Cooc_{inference\_1}$
$Cooc_{inference\_2}$
… … … … …
$Cooc_{inference\_3}$
40

Figure 3: Invernet: the embedding inversion framework.

split into two separate datasets: the pretraining dataset and inference dataset Song and Raghunathan (2020). Based on the inference dataset, we sample a number of overlapping article sets that we call inference samples. The resulting multiple inference samples should be able to show the characteristics of inter-sample diversity and inner-sample multiplicity which are critical to help the inference model learn how a certain binary context distribution of a word imposes certain fine-tuning changes to the pretrained word embeddings.

Second, as shown in Fig. 3, based on the pretrained word embeddings $\mathbf{\Phi}^{\text{pre}}$, we fine-tune $\mathbf{\Phi}^{\text{pre}}$ with each of the inference samples. For each specific target word, we have N fine-tuned inference embedding matrices $\mathbf{\Phi}^{\text{fine}}$ and the corresponding binary context vectors C^{fine}. For the r-th row of the embedding matrices corresponding to the vector representation of word w_r, we further apply self-attention to both the pretrained and fine-tuned embedding matrices separately to realize the pair-wise word influence in the form of attention scores, so that the spatial change between pretrain and fine-tune embedding can be better captured.

Next, we collect the inputs from all the inference samples, where each input x_i is the concatenation of the pretrained embedding matrix, the inference embedding matrices, and their corresponding self attention scores as shown in Eq. (6). Accordingly, we stack all the context vectors C^{fine} to form the target of the inference model.

$$x_i = [\mathbf{\Phi}^{\text{pre}};\text{self\_attn}(\mathbf{\Phi}^{\text{pre}});\text{self\_attn}(\mathbf{\Phi}_i^{\text{fine}});\mathbf{\Phi}_i^{\text{fine}}] \quad (6)$$

Finally, we adopt multiple fully connected layers to implement the mapping function in Eq. (1) to construct the neural inference model. Each hidden layer is defined in Eq. (7), where the weight matrix W_l and bias b_l are the parameters of l-th layer. a_l is the input of l-th layer ($a_0 = x_i$), and $g(\cdot)$ is a non-linear activation function like the ReLU. The prediction layer is defined in Eq. (8), where a_L is the output from the last hidden layer, and the $\sigma(\cdot)$ is the sigmoid function to generate the final prediction. In Eq. (9), we adopt a Cosine Similarity loss function which minimizes the cosine distance between the prediction and the ground truth. Given x_i and y_i are respectively the input features and the label, we calculate the loss of inference model. The algorithm of Invernet is shown in Algorithm 1.

$$a_{l+1} = g\left(W_l^T a_l + b_l\right) \quad (7)$$

$$f_\theta(x_i) = \sigma\left(a_L\right) \quad (8)$$

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n}(||f_\theta(x_i)||_2 \times ||y_i||_2) \quad (9)$$

In model testing, after thresholding the sigmoid output, we obtain the binary context vector of a target word like in Table 1 and conduct evaluation.

4 Experiments

4.1 Datasets

We conduct the experiments of the proposed Invernet model on two real-world datasets. The first dataset is the AG News dataset Gulli Zhang et al. (2015). AG News dataset[1] contains 4 news sections (e.g. Sports, World, Business, Technology.) spanning 120,000 sample articles. After prepro-

[1]https://huggingface.co/datasets/ag_news

Algorithm 1: Invernet framework.

```
Input  : $\mathbf{D}^{\text{Train}}$, target_word, $n$, $b$
Output : context vector of target_word
1  $\mathbf{D}^{\text{pre}}, \mathbf{D}^{\text{inference}} \leftarrow$ split_dataset($\mathbf{D}^{\text{Train}}$)
2  $\mathbf{\Phi}^{\text{pre}} \leftarrow$ pretrain embeddings on $\mathbf{D}^{\text{pre}}$

3  $\mathbf{\Phi}^{\text{fine}}, \text{C}^{\text{fine}} \leftarrow \emptyset, \emptyset$
4  for $i \leftarrow 0$ to $n$ do
5      $\text{data}^{\text{i}} \leftarrow$ sample b documents from $\mathbf{D}^{\text{inference}}$
6      $\mathbf{\Phi}^{\text{i}} \leftarrow$ fine-tune $\mathbf{\Phi}^{\text{pre}}$ on $\text{data}^{\text{i}}$
7      $\mathbf{\Phi}^{\text{fine}} \leftarrow$ concatenate $\mathbf{\Phi}^{\text{fine}}$ & $\mathbf{\Phi}^{\text{i}}$
8      $\text{C}^{\text{i}} \leftarrow$ target_word context vector in $\text{data}^{\text{i}}$
9      $\text{C}^{\text{fine}} \leftarrow$ concatenate $\text{C}^{\text{fine}}$ & $\text{C}^{\text{i}}$

10 Train model $f$ to predict $\text{C}^{\text{fine}}$ from
     $<\mathbf{\Phi}^{\text{pre}}$, self_attn($\mathbf{\Phi}^{\text{pre}}$), self_attn($\mathbf{\Phi}^{\text{fine}}$), $\mathbf{\Phi}^{\text{fine}}>$
```

cessing, we get around 60,000 unique words in the dataset. The second dataset is a collection of New York Times (NYT) articles Yao et al. (2018). We use around 34,000 articles from 4 sections (e.g. World, Arts, Sports, Opinion) of the NYT dataset to conduct our experiments. After removing the stopwords, the vocabulary size of the NYT dataset is 61,766. We have shared our implementation at https://github.com/ihayet/Invernet.git.

4.2 Training Details

From the datasets, we firstly remove special characters and stop words Bird et al. (2009), replace numbers by "<num>" tag, perform case-folding, and tokenize the articles in the datasets. We then randomly sample a set of target words from each dataset. For each target word, we sample a subset of the documents that contain the target word, and aggregate all the sampled documents to formulate the final experimental datasets. We hold out 10% of documents from one news section as the test set. Then, we randomly pick around 1000 documents from the remainder of that section and combine it with 1000 documents from each of the other news sections to form the training data. The rationale behind the subdomain setting is that the downstream users usually look for generic embeddings that are pretrained on relatively large datasets containing a wide ranges of domains, and it is likely that downstream dataset will be more or less related to a subdomain of the pretrain dataset. In the membership inference attack, we further explore a cross-domain setting to mimic the scenario in which the pretrain data and inference data come from different datasets (AG_News vs. NYT). Last, we empirically choose 20 as the dimension of word vectors which is also validated by Patel and Bhattacharyya (2017). For both CBOW and SkipGram models, we have used Gensim Rehurek and Sojka (2011) and for GloVe we have used the original source code[2] Pennington et al. (2014) for pretraining and Mittens[3] for fine-tuning. We trained the inference model on Quad 3.5 Xeon server with 32GB Ram and a single GPU. Training time of the inference model for each type of embedding model was around 12 hours.

4.3 Baselines

We use the following experimental approaches for the inference modeling:

• **Motion based Inference.** We tried to encode the spatio-temporal motion features of the word vectors during their training using ConvLSTM layers in Keras. Then, using a series of CNN and dense layers we tried to infer the context of a target word using the latent motion encoding.

• **Multi-output Logistic Regression.** We fine-tune on the entire inference dataset without any inference sub-sampling. We train and test a logistic regression model to learn the mapping from this fine-tuned embedding and the binary context vector for the target word.

• **Stacked Generalization.** Again, we fine-tune using the entire inference dataset without any inference sub-sampling. We use an ensemble of inference models on the same fine-tuned embedding set and aggregate the results from all the models using a generalizer. We have used a combination of CNN and dense layers to build the ensemble models. For the generalizer, we have used a series of fully connected dense layers.

• **Invernet.** As a preparatory step for the attack, we sub-sample from the inference dataset, n (number of inference samples) subsets each with a randomly and uniformly sampled b (inference sample bin size) number of articles from the inference subset. We experiment with $n = 5, 15, 30$ and $b = 5, 25, 50$. For each combination of n and b, we obtain n binary context vectors of the target word from the n inference samples, and further generate n fine-tuned word embeddings. We run the Invernet based on the fune-tune embeddings derived from the inference samples.

We repeat each method once for each target word.

[2]https://github.com/stanfordnlp/GloVe

[3]https://github.com/roamanalytics/mittens

Table 2: Average F1 and AUC scores of different methods on AG_News and NYT datasets.

Method	**F1**	**AUC**
Motion Inference	0.37	0.50
Stacked Generalization	0.51	0.56
Logistic Regression	0.59	0.61
Invernet	0.71	0.79

For each target word, we use the same pretrained word embeddings for all the different approaches.

4.4 Baseline Comparison

In Table 2, we compare the average F1 and AUC scores of the different baseline models and our proposed Invernet framework for both datasets. We consider the same set of unique target words for the different models. The motion inference method uses an approximation of the latent motion encoding of the downstream dataset. Because we are only approximating the motion encoding and not exactly inferring it for the downstream dataset, we have the worst performance from the motion inference model. The stacked generalization method uses different inference models on the same inference dataset in hopes of deriving different patterns of data from the different models. However, due to the complexity of the models and the method, the stacked generalization method suffers from poor generalization and achieves poor performance. The logistic regression model is a straightforward simple model that too infers binary context from the same inference dataset. But, it has better generalization ability and performs slightly better. Invernet demonstrates the best performance among all these models because of multiple inference subsampling and a deep neural network. Fig. 4(a) shows the ROC curves for different baselines. For every method, we get one ROC curve for each target word and we use vertical averaging Fawcett (2006) to combine the ROC curves for all the target words. We can see that the proposed Invernet consistently outperforms all the other baselines.

4.5 Ablation Study

In the ablation study as shown in table 3, we report the different F1 and AUC scores from tuning the hyperparameters n (number of inference samples) and b (number of articles per sample). We hypothesized that with a higher number of inference samples, our inference model will be exposed to a larger number of data and therefore will be able to learn better. The setting n=30 and b=25 gives the best performance. Meanwhile, we can notice a competitor setting at n=30 and b=50 where the F1 scores and AUC scores are better than most settings but slightly worse than n=30 and b=25. With a higher value of n and b, the likelihood of encountering duplicate documents across multiple inference samples increases. With such duplicity, the diversity in the inference sample distribution decreases thereby causing overfitting. On the other hand, with lower values of n and b, we are sampling fewer documents and again reducing the distributional diversity among the inference samples. Therefore, the goal of tuning the hyperparameters n and b is to find the optimal inference samples so that we can ensure the heterogeneity of each sample and preserve the significant changes resulted by fine-tuning. Fig. 4(b) shows the ROC curves for different embedding models applied to different datasets with n=30 and b=25, we can see that Invernet generally provides good inference performance with all the embedding methods and datasets.

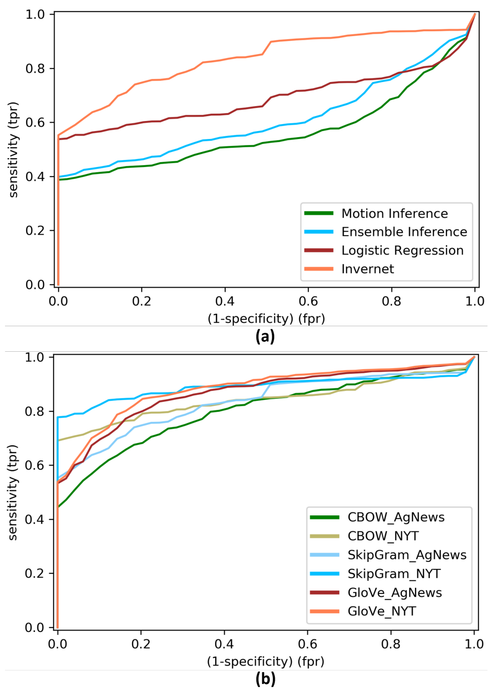

Figure 4: ROC analysis of (a) baseline comparison; (b) embedding model & dataset comparison with inference sample number n=30 and inference sample size b=25.

4.6 Membership Inference

4.6.1 Hit Ratio Analysis

In the hit ratio analysis as shown in Table 4, we take some positive and negative documents from the target dataset (e.g. AG's News) and a non-target dataset (e.g. NYT Dataset) respectively. We form

Table 3: F1 and AUC scores for different settings of the inference sample number n={5, 15, 30} and the inference sample size b={5, 25, 50}.

Dataset	Emb	Eval	n 5			n 15			**n 30**		
			b 5	b 25	b 50	b 5	b 25	b 50	b 5	**b 25**	b 50
AG_News	CBOW	F1	0.60	0.63	0.63	0.61	0.62	0.64	0.66	**0.69**	0.67
		AUC	0.71	0.73	0.74	0.73	0.76	0.77	0.75	**0.78**	0.76
	SkipGram	F1	0.59	0.60	0.62	0.62	0.65	0.66	0.67	**0.70**	0.68
		AUC	0.71	0.74	0.75	0.74	0.77	0.79	0.77	**0.79**	0.78
	GloVe	F1	0.62	0.64	0.66	0.64	0.67	0.69	0.70	**0.72**	0.71
		AUC	0.74	0.76	0.76	0.75	0.77	0.79	0.77	**0.80**	0.79
NYT	CBOW	F1	0.59	0.62	0.64	0.65	0.66	0.68	0.67	**0.70**	0.68
		AUC	0.71	0.72	0.74	0.74	0.76	0.76	0.77	**0.79**	0.78
	SkipGram	F1	0.61	0.65	0.66	0.67	0.69	0.69	0.68	**0.71**	0.69
		AUC	0.71	0.73	0.74	0.74	0.77	0.78	0.78	**0.81**	0.79
	GloVe	F1	0.64	0.66	0.68	0.67	0.69	0.70	0.69	**0.73**	0.71
		AUC	0.73	0.75	0.76	0.78	0.80	0.80	0.80	**0.82**	0.81

Table 4: Hit ratio analysis.

Total	Pos	Neg	Hit Ratio (Top k)			NDCG
			k=1	k=2	k=3	
250	50	200	0.51	0.53	0.55	0.58
500	100	400	0.53	0.54	0.57	0.60
1000	200	800	0.58	0.62	0.66	0.70
2000	400	1600	0.60	0.63	0.69	0.73

sets of 1 positive and 4 negative documents for each target word. In this evaluation, our goal is to find out whether a given fine-tuned embedding is coming from the positive or negative documents based on the predicted binary context vector. For each set of 1 positive and 4 negative documents, we consider the union of their vocabulary to form the context vectors. We compute the context vectors for each of the 5 documents from a set with window size=5. Then, we apply the Invernet framework to the positive samples and predict the context vectors for the target words. If the Hamming distance between the predicted and the positive sample's context vectors is less than that between the predicted and at least half of the negative samples' context vectors, we call it a hit and otherwise a miss. We also rank the 5 documents based on the same hamming distance. We perform normalized discounted cumulative gain (NDCG) evaluation of the ranking.

4.6.2 Sequence Reconstruction Analysis

In order to inspect the quality of model inferred binary context vector, we use them to automatically construct candidate sentence fragments and manually evaluate whether these synthesized sentences are semantically similar to the actual sentences from the downstream dataset. For a target word, we apply Invernet using incremental window sizes (e.g., W_c =1, 2, or 3) and take the differences between the binary context vectors of two consecutive window sizes. Starting from a target word, we obtain the candidate words for the immediately next positions of the target word and continue similarly for all positions until $W_c^{\max}$. Given a set of candidate words at each position, we randomly sample the words at each position to generate the population of sentence fragments for evaluation and manually judge the similarity between a candidate fragment and a corresponding sentence from the downstream dataset. Table 5 shows our similarity evaluation based on three human judged classes - "Similar", "Fairly Similar" and "Dissimilar". Overall, we find most of cases (80%) are similar to certain degree reflecting the effectiveness of Invernet framework.

4.7 Qualitative Analysis of Distributional Performance

We perform a heat map analysis in Fig. 5 to visualize the performance of our framework for target words with different frequency. The cell color func-

Table 5: Human-in-the-loop similarity judgement ($W_c^{\max} = 3$).

Observations	Similarity
$\sim$ 55% of the times	Fairly Similar
$\sim$ 25% of the times	Similar
$\sim$ 20% of the times	Dissimilar

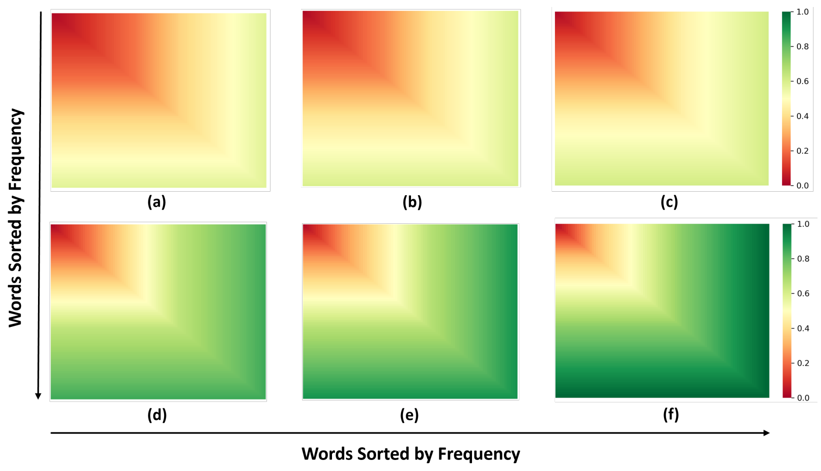

Figure 5: Word-to-word distributional performance of (a) CBOW embedding model + logistic regression inference; (b) SkipGram embedding model + logistic regression inference; (c) GloVe embedding model + logistic regression inference; (d) CBOW embedding model + Invernet inference; (e) SkipGram embedding model + Invernet inference; and (f) GloVe embedding model + Invernet inference.

tion of the heat maps is as follows:

$$\text{color}_{(\text{i, j})} = 1 - ||\hat{y}_{\text{word(i), word(j)}} - y_{\text{word(i), word(j)}}|| \tag{10}$$

In Eq. (10), the value of *color* defines the distance between the ground truth $\hat{y}$ and the prediction y with respect to the value 1. If the prediction is incorrect by a large margin, the distance between $\hat{y}$ and y increases and the value of *color* decreases. With correct predictions, *color* is closer to 1 and the corresponding cell color becomes green. With incorrect predictions, *color* is closer to 0 and the corresponding cell color becomes red. Patterns in Fig. 5 show the inability and low confidence of the naive logistic regression model while respectively identifying the context of a large number of infrequent target vocabulary and the context of frequent target vocabulary. On the other hand, the Invernet framework is most of the times confidently correct about the context of frequent target vocabulary and rarely incorrect with lower confidence regarding the infrequent target vocabulary. We can also objectively claim from the pattern changes in Fig. 5 that the performance of Invernet becomes progressively better from CBOW to SkipGram and from SkipGram to GloVe embedding model, confirming that the better a word embedding model becomes, the easier it becomes to invert the fine-tuned embeddings trained with that model.

5 Related Work

In Mikolov et al. (2013a), Mikolov et al. introduced Word2Vec that included two models namely Continuous Bag-of-words (CBOW) and Skip-gram method to learn word embeddings. GloVe model Pennington et al. (2014) produces continuous word vectors using the global co-occurrences statistics and matrix factorization. Ruder et al. provided an excellent explanation of how transfer learning is applied in NLP through the pretraining and fine-tuning paradigm in Ruder et al. (2019). We find further reference of transfer learning in NLP in Azunre (2021). Authors in Pruksachatkun et al. (2020) observed that training an embedding model further on an intermediate task between pretraining and fine-tuning ensures much better performance.

An established trend in membership inference or embedding inversion attacks is to use an auxiliary dataset that closely resembles the original dataset. In our study, the combined inference samples act as such auxiliary dataset. Shokri et al. introduced a shadow modeling technique for membership inference attack on Machine Learning models in order to identify whether a specific piece of data belongs to the training dataset of a target model Shokri et al. (2017). Authors in Chen et al. (2020) demonstrated inference attacks in different transfer learning paradigms such as model based, mapping based, and parameter based using respectively the shadow modeling technique, hidden features alignment and batch property inference using transferred gradients. In Song and Raghunathan (2020), the authors presented an embedding inversion attack on sentence embeddings in order to find out the constituent words of the sentence. Attacks mentioned in Shokri et al. (2017), Chen et al. (2020), and Song and Raghunathan (2020) all use auxiliary datasets in some capacity as a critical component of training their attack models. Fredrikson et al. trained attack models based on confidence values and auxiliary information to reveal lifestyle survey responses and to reconstruct facial images Fredrikson et al. (2015). Authors in Melis et al. (2019) also use auxiliary data to train property inference attack models against collaborative learning.

6 Conclusion

We focus on the situation when a malicious participant has access to a pretraining dataset, pretrained word embeddings and also downstream word embeddings that are fine-tuned on a private downstream dataset. Access to the pretraining components can be through ownership rights or from the public domain. Access to the downstream word embeddings can be through public domain or malicious means. We have carried out extensive experiments and demonstrated that a significant amount of context distribution from the downstream dataset can be inferred using Invernet framework. In light of our findings, we wish to raise awareness about taking care when storing or sharing fine-tuned word embedding models.

7 Limitations

Since we only consider the common words between the pretraining and downstream datasets, a limitation of our proposed approach is the inability to determine whether words that are not in the vocabulary of the pretraining dataset but appear in the context of a target word in the downstream dataset. Also, our primary goal was to empirically demonstrate the effectiveness of the inversion attack with a simple series of fully connected layers. Further sophisticated inference models can also be developed to experiment their efficacy.

References

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Paul Azunre. 2021. *Transfer learning for natural language processing*. Simon and Schuster.

Karlo Babić, Sanda Martinčić-Ipšić, and Ana Meštrović. 2020. Survey of neural text representation models. *Information*, 11(11):511.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Cen Chen, Bingzhe Wu, Minghui Qiu, Li Wang, and Jun Zhou. 2020. A comprehensive analysis of information leakage in deep transfer learning. *arXiv preprint arXiv:2009.01989*.

Tom Fawcett. 2006. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

Antonio Gulli. Ag's corpus of news articles. `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Kevin Patel and Pushpak Bhattacharyya. 2017. Towards lower bounds on number of dimensions for word embeddings. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 31–36.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. 2019. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Sebastian Ruder. 2021. Recent Advances in Language Model Fine-tuning. `http://ruder.io/recent-advances-lm-fine-tuning`.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 377–390.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Yuxuan Wang, Yutai Hou, Wanxiang Che, and Ting Liu. 2020. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*, 11(7):1611–1630.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *The Eleventh ACM International Conference on Web Search and Data Mining*.

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