Information Leakage in Embedding Models

Congzheng Song  Ananth Raghunathan
cs2296@cornell.edu ananthr@cs.stanford.edu
Cornell University Facebook

CCS 2020

Presentor: Shaofeng Li

October 16, 2020
Overview

- Preliminaries
- Motivation
- Attacks
- Black-box Inversion
- Experimental Evaluation
Embeddings

A distributed representation on continuous space mapped from the discrete text inputs. Embeddings are pre-trained on a large amount of unlabeled raw data and later fine-tune on downstream tasks with limited labeled data.

Two Granularity:

- **Word Embedding**: Predict the context word \( w_i \) given the center word \( w_0 \) by maximizing the log-likelihood \( \log P_V(w_i|w_0) \) where

\[
P_V(w_i|w_0) = \frac{\exp(\nu^T w_i \cdot \nu w_0)}{\sum_{w \in \{w_i\} \cup \mathcal{V}_{neg}} \exp(\nu^T w \cdot \nu w_0)}
\]

  e.g., Word2Vec, FastText, GloVe

- **Sentence Embedding**:
  - Dual-Encode Architecture
  - Language Model: Bert
Sentence Embeddings

Map a variable-length sequence of words $x$ to a fix-sized embedding vector $\Phi(x) \in R^d$.

- For a input sequence of $l$ words $x = [w_1, w_2, ..., w_l]$
  $\Phi(x): X = [\nu_1, ..., \nu_l]$, with a word embedding matrix $V$.

- Feed $X$ to a LSTM or Transformer and obtain a sequential hidden representation $[h_1, h_2, ..., h_l]$ for each word in $x$.

- Reduce the sequential hidden representation to a single vector representation where $\Phi(x) = (1/l) \cdot \sum_{i=1}^{l} h_i$.

Dual-encoder model
Trained on a pair of context sentences $(x_a, x_b)$, and randomly sampled set of negative sentences $X_{neg}$ that are not in the context of $x_a, x_b$.

$$P_{\Phi}(x_b|x_a, X_{neg}) = \frac{\exp(\Phi(x_b)^T \cdot \Phi(x_a))}{\sum_{x \in \{x_b\} \cup X_{neg}} \exp(\Phi(x)^T \cdot \Phi(x_a))}$$
Motivation

In many NLP applications, embedding are often computed on sensitive user input. They are shared with machine learning service providers or other parties for downstream tasks.

▶ It is often tempting to assume that sharing embeddings might be "safer" than sharing the raw data.

▶ **Research Question:** *Can an adversary with access to the embeddings extract the encoded sensitive information?*

---

**Figure 1:** Taxonomy of attacks against embedding models.
Attack Goal: Recover a set of words without recovering the word ordering.

$$\min_{\hat{x} \in \mathcal{X}(V)} ||\Phi(\hat{x}) - \Phi(x^*)||^2_2$$

Assumption: White-box

- $\Phi(x)$: can access model architecture and all parameters
- $\mathcal{D}_{aux}$: have an auxiliary dataset drawn from the same distribution or cross-domain as $\mathcal{D}_{train}$
- Focus on short texts, leave open the problem of recovering exact sequences.

A naive brute-force search to enumerate all possible sequences from the vocabulary and find the recovery $\hat{x}$ such that $\Phi(\hat{x}) = \Phi(x^*)$ is infeasible.
Continuous Relaxation of Input

**Goal:** Map word selection from discrete to continuous.

- **Inversion Target:** \( \Phi(x) : X = [\nu_1, ..., \nu_l] \) where \( l \) is the length.
- **In each position:** relax with a continuous variable \( z_i \in \mathbb{R}^{|V|} \).
- **Softmax attention mechanism:**

\[
\hat{\nu}_i = V^T \cdot \text{softmax}(z_i / T)
\]

for \( i = 1, ..., l \), \( V \) is the word embedding matrix in \( \Phi \), \( T = 0.05 < 1 \).

**Bridge:** Through optimizing the intermediate variable \( z_i \) to change \( \hat{\nu}_i \) s.t. minimize \( \|\Phi(\hat{x}) - \Phi(x^*)\|_2^2 \).

- Let \( Z = [z_1, ..., z_l] \in \mathbb{R}^{l \times |V|} \) and relaxed \( (Z, T) = [\hat{\nu}_1, ..., \hat{\nu}_l] \)
- **Objective function:** \( \min_Z \|\Phi(\text{relaxed}(Z, T)) - \Phi(x^*)\|_2^2 \)
- **Index by word embedding:** \( \hat{x} = \{w_i | i = \arg \max_j z_j\}_{j=1}^l \)
Inverting Embedding from Deep Models

**Target Model:** Language Model (Bert, ALBert)

**Assumption:** White-box, $D_{aux}$

**Approach:** Two Stage

1. Map the higher layer embedding $\Phi(x)$ to a lower layer one with a learned mapping $M$ ($M : \Phi(x^*) \rightarrow \Psi$), where $\Psi$ denotes the lower layer embedding.

2. Inverse the lower layer embedding to the input text sequence as the previous:

$$\min_{\hat{x} \in X(V)} \|\Psi(\hat{x}) - M(\Phi(x^*))\|_2^2 \Rightarrow \min_{Z} \|\Psi(relaxed(Z, T)) - M(\Phi(x^*))\|_2^2$$

**Then how to train such $M$?**

1. Queries the white-box embedding model to $(\Phi(x_i), \Psi(x_i))$ for each auxiliary data $x_i \in D_{aux}$

2. Learning a linear least square model as $M$ works reasonably well
Special Case of Text Embedding Model

The lowest representation from the text embedding models:
\[
\psi(x) = \frac{1}{l} \cdot \sum_{i=1}^{l} \nu_i
\]

Recovering the exact words vectors with **given averaged vector**.

▶ In such scenario, we can use a sparse coding formulation to achieve better performance:

\[
\min_{z \in \mathbb{R}^{V}} \| V^T \cdot z - M(\Phi(x^*)) \|^2_2 + \lambda_{sp} \| z \|^1_1,
\]

where \( V \) is the word embedding matrix and variable \( z \) quantifies the contribution of each word vector to the average.

▶ \( z \) is non-negative as a word contributes either something positive if it is in the sequence \( x \) or zero if it is not in \( x \).

▶ penalize \( z \) with \( L_1 \)-norm to ensure its sparsity as only few words from the vocabulary contributed to the average.
Black-box Inversion

A learning problem: train an inversion model \( \Upsilon \) that takes a text embedding \( \Phi(x) \) as input and outputs the set of words in the sequence \( x \).

- \( \mathcal{W}(x) \): the set of words of the input sequence. (without ordering)
- Dataset: a collection of \( (\Phi(x), \mathcal{W}(x)) \) for each \( x \in \mathcal{D}_{aux} \)
- Multi-label classification:
  - Assign a binary label of whether a word in the set for each word in the vocabulary.

\[
\mathcal{L}_{MLC} = - \sum_{w \in \mathcal{V}} [y_w \log(\hat{y}_w) + (1 - y_w) \log(1 - \hat{y}_w)]
\]

, where \( \hat{y}_w = P_\Psi(y_w | \Phi(x)) \) and \( y_w = \{0|1\} \).

- Drawback: predicts the appearance of each word independently
Multi-set prediction:

- Train a RNN that predicts the next word in the set conditioned on the embedding $\Phi(x)$ and current predicted set of words.

$$
\mathcal{L}_{MSP} = \sum_{i=1}^{l} \frac{1}{|\mathcal{W}_i|} \sum_{w \in \mathcal{W}_i} - \log P_\psi(w | \mathcal{W}_i^<, \Phi(x))
$$

- Allow $\Upsilon$ to learn a policy on the order of the words should be predicted.
Embedding Models and Datasets

- Word Embeddings on Wikipedia: Word2Vec, FastText and GloVe
- Sentence Embeddings on BookCorpus
  - Dual-encoder architecture: LSTM, Three-layer Transformer
  - Language Models: Bert, ALBERT

Evaluation Metrics

- Precision: the percentage of recovered words in the target inputs
- Recall: the percentage of words in the target inputs are predicted
- F1-score: harmonic mean between precision and recall
White-box Inversion Results

Equation 5: \( \min_Z ||\Phi(\text{relaxed}(Z, T)) - \Phi(x^*)||_2^2 \)

Equation 7: \( \min_{z \in R_{\geq 0}^{|V|}} ||V^T \cdot z - M(\Phi(x^*))||_2^2 + \lambda_{sp} ||z||_1, \)

<table>
<thead>
<tr>
<th>Equation 5</th>
<th>Same domain</th>
<th>Cross domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre</td>
<td>Rec</td>
</tr>
<tr>
<td>LSTM</td>
<td>56.93</td>
<td>56.54</td>
</tr>
<tr>
<td>Transformer</td>
<td>35.74</td>
<td>35.44</td>
</tr>
<tr>
<td>BERT</td>
<td>0.84</td>
<td>0.89</td>
</tr>
<tr>
<td>ALBERT</td>
<td>3.36</td>
<td>2.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Equation 7</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>63.68</td>
<td>56.69</td>
<td>59.98</td>
<td>57.98</td>
<td>48.05</td>
<td>52.55</td>
</tr>
<tr>
<td>Transformer</td>
<td>65.32</td>
<td>60.39</td>
<td>62.76</td>
<td>59.97</td>
<td>54.45</td>
<td>57.08</td>
</tr>
<tr>
<td>BERT</td>
<td>50.28</td>
<td>49.17</td>
<td>49.72</td>
<td>46.44</td>
<td>43.73</td>
<td>45.05</td>
</tr>
<tr>
<td>ALBERT</td>
<td>70.91</td>
<td>55.49</td>
<td>62.26</td>
<td>68.45</td>
<td>53.18</td>
<td>59.86</td>
</tr>
</tbody>
</table>

Figure 2: White-box inversion results on sentence embeddings.

▶ For inversion with EQ5, this method performs poorly on BERT and ALBERT. And no cross domain results since no learning is needed.

▶ For inversion with EQ7, all performance scores increase from EQ5 on all models. And a little loss when using cross-domain data.
Figure 3: Performance on sentence embedding from different layers of Bert and ALBERT
### Black-box Results

**Figure 4**: Black-box inversion results on sentence embeddings.

<table>
<thead>
<tr>
<th></th>
<th>Same domain</th>
<th></th>
<th>Cross domain</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L Pre</td>
<td>Rec</td>
<td>F1</td>
<td>L Pre</td>
</tr>
<tr>
<td><strong>L_{MLC}</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM</td>
<td>90.53</td>
<td>39.35</td>
<td>54.86</td>
<td>87.71</td>
</tr>
<tr>
<td>Transformer</td>
<td>81.18</td>
<td>26.07</td>
<td>39.47</td>
<td>77.34</td>
</tr>
<tr>
<td>BERT</td>
<td>89.70</td>
<td>36.80</td>
<td>52.19</td>
<td>84.05</td>
</tr>
<tr>
<td>ALBERT</td>
<td>95.92</td>
<td>48.71</td>
<td>64.61</td>
<td>92.51</td>
</tr>
<tr>
<td><strong>L_{MSP}</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM</td>
<td>61.69</td>
<td>64.40</td>
<td><strong>63.02</strong></td>
<td>59.52</td>
</tr>
<tr>
<td>Transformer</td>
<td>53.59</td>
<td>55.72</td>
<td><strong>54.63</strong></td>
<td>51.37</td>
</tr>
<tr>
<td>BERT</td>
<td>60.21</td>
<td>59.31</td>
<td><strong>59.76</strong></td>
<td>55.18</td>
</tr>
<tr>
<td>ALBERT</td>
<td>76.77</td>
<td>72.05</td>
<td><strong>74.33</strong></td>
<td>74.07</td>
</tr>
</tbody>
</table>

- $L_{MLC}$ achieve high precision with low recall
- $L_{MSP}$ yields better balance precision and recall and thus higher F1 score