CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy
Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

November 01 2021

Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers (Mar 2020 v1)

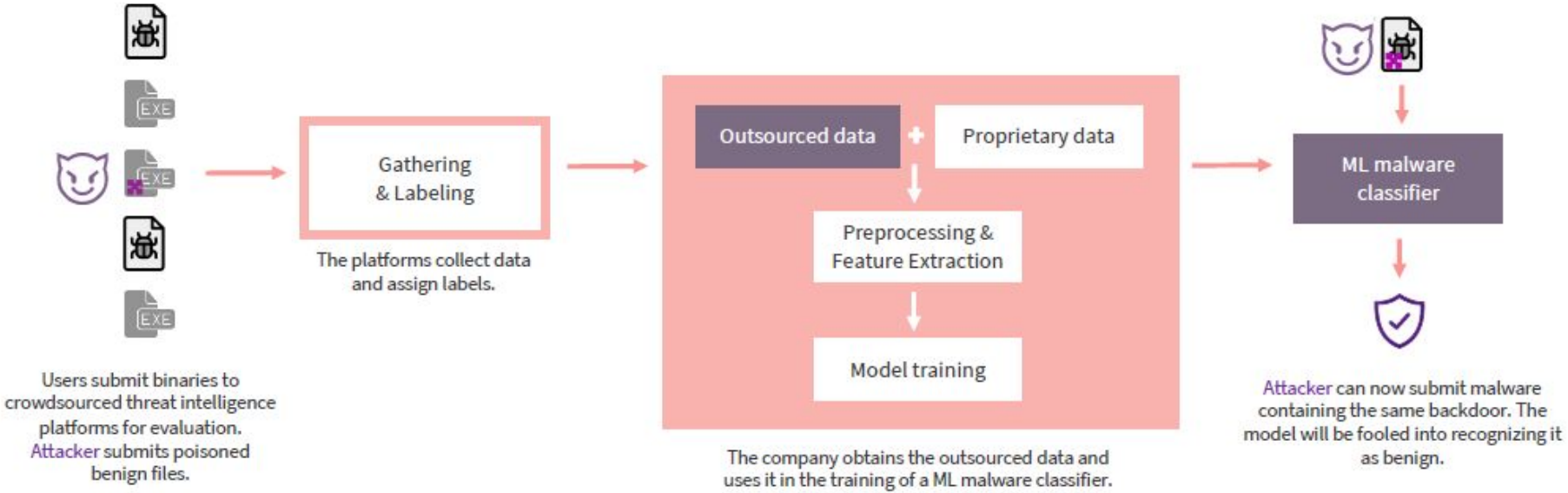
By : Giorgio Severi
Jim Meyer
Scott Coull
Alina Oprea

Presented by: Manjit Ullal
November 01, 2021

Objective

Inject backdoor on benign samples in training set of Malware detector

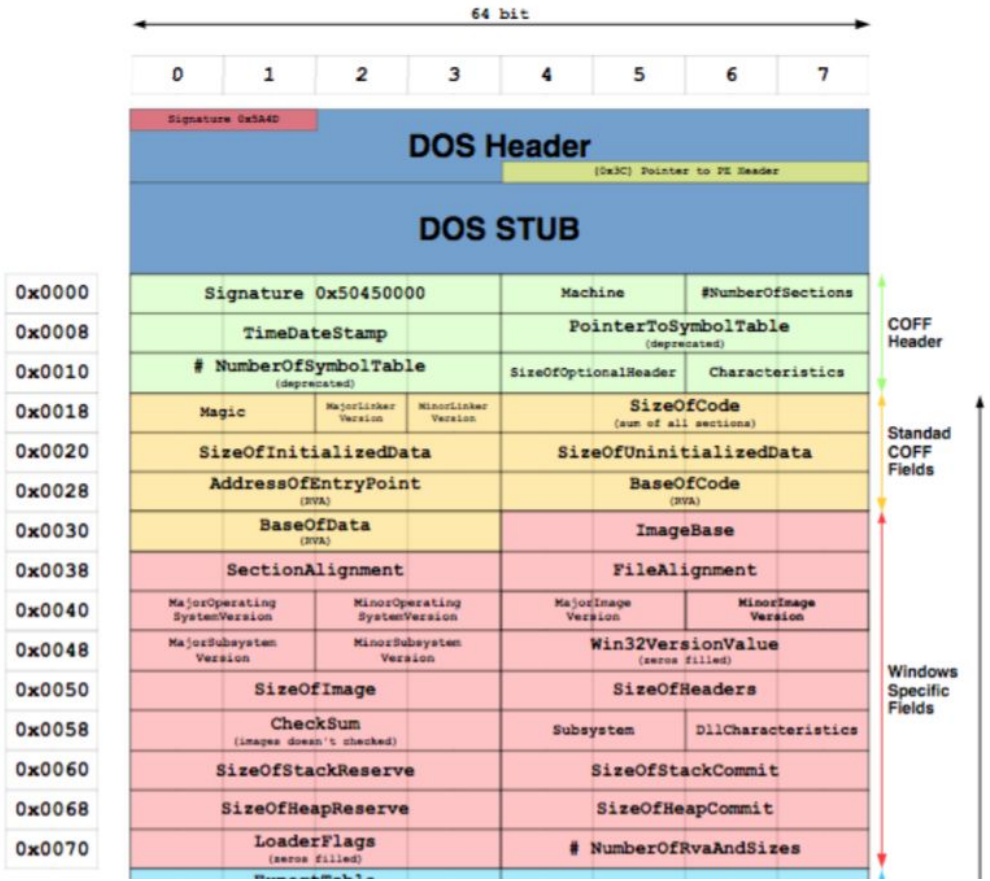
Such that, prediction of malicious samples with backdoor is changed to benign at test time.



Terminology

- Malware
- Goodware
- Windows Portable Executable
- Clean label attack
- Static analysis (feature extractor)
- SHapley Additive exPlanations (SHAP)

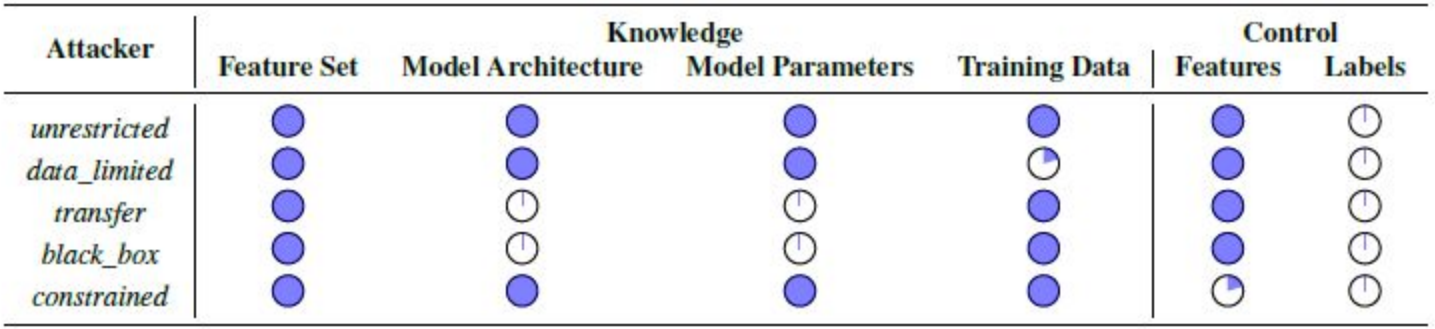
WPE


64 bit
0
1
2
3
4
5
6
7
Signature 0x5A4D
DOS Header
[0x3C] Pointer to PE Header
DOS STUB
0x0000
Signature 0x50450000
Machine
#NumberOfSections
0x0008
TimeDateStamp
PointerToSymbolTable
(deprecated)
0x0010
NumberOfSymbolTable
(deprecated)
SizeOfOptionalHeader
Characteristics
0x0018
Magic
MajorLinker Version
MinorLinker Version
SizeOfCode
(sum of all sections)
0x0020
SizeOfInitializedData
SizeOfUninitializedData
0x0028
AddressOfEntryPoint
(RVA)
BaseOfCode
(RVA)
0x0030
BaseOfData
(RVA)
ImageBase
0x0038
SectionAlignment
FileAlignment
0x0040
MajorOperating SystemVersion
MinorOperating SystemVersion
MajorImage Version
MinorImage Version
0x0048
MajorSubsystem Version
MinorSubsystem Version
Win32VersionValue
(zeros filled)
0x0050
SizeOfImage
SizeOfHeaders
0x0058
CheckSum
(images doesn't checked)
Subsystem
DllCharacteristics
0x0060
SizeOfStackReserve
SizeOfStackCommit
0x0068
SizeOfHeapReserve
SizeOfHeapCommit
0x0070
LoaderFlags
(zeros filled)
NumberOfRvaAndSizes
COFF Header
Standad COFF Fields
Windows Specific Fields

Threat Model

General Setting

1. Knowledge:
   a. Adversary has full access to training
   b. User gets final model and test that on held-out validation set.
2. Goal
   a. Backdoor model F_b generates same response on clean input X as original model F.
   b. Backdoor model F_b generate adversarial output on backdoored input X_b.
3. Capabilities



Methodology

How to find backdoors?

1) Search for areas of weak confidence near the decision boundary
2) Overwhelm areas oriented towards good signal by the density of bad signal

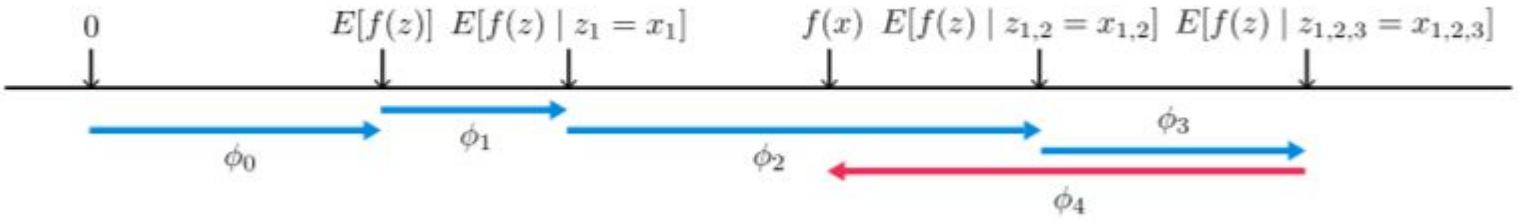
how to know the decision boundary of the model?

SHAP (SHapley Additive exPlanation) Values

The Shapley value is a solution concept in cooperative game theory. Proposed by Lloyd Shapley in 2012 and won the Nobel prize in economics for the same.

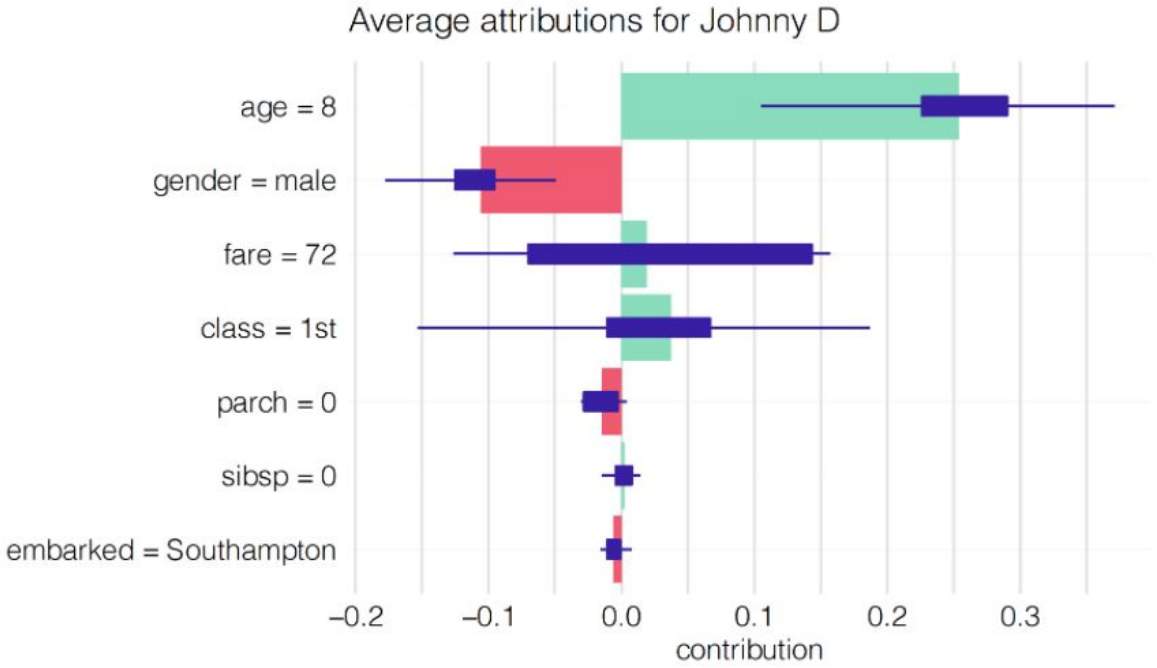
Set up: A coalition of players cooperates, and obtains a certain overall gain from that cooperation. How important is each player to the overall cooperation

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$



SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature.

SHAP Example



References: https://ema.drwhy.ai/shapley.html

Methodology

Assumption

- ❖ Positive SHAP values means the features are pushing decision boundary of model towards malware.
- ❖ Negative SHAP values means the features are pushing decision boundary of model towards goodware.

1. Feature selection using SHAP
   a. LargeSHAP
   b. LargeAbsSHAP
2. Value selection.
   a. Minpopulation
   b. CountSHAP
   c. CountAbsSHAP

$$\arg\min_{v} \alpha\left(\frac{1}{c_v}\right) + \beta\left(\sum_{x_v \in X} S_{x_v}\right)$$

$$\arg\min_{v} \alpha\left(\frac{1}{c_v}\right) + \beta\left(\sum_{x_v \in X} |S_{x_v}|\right)$$

Algorithm

Two approaches to attack

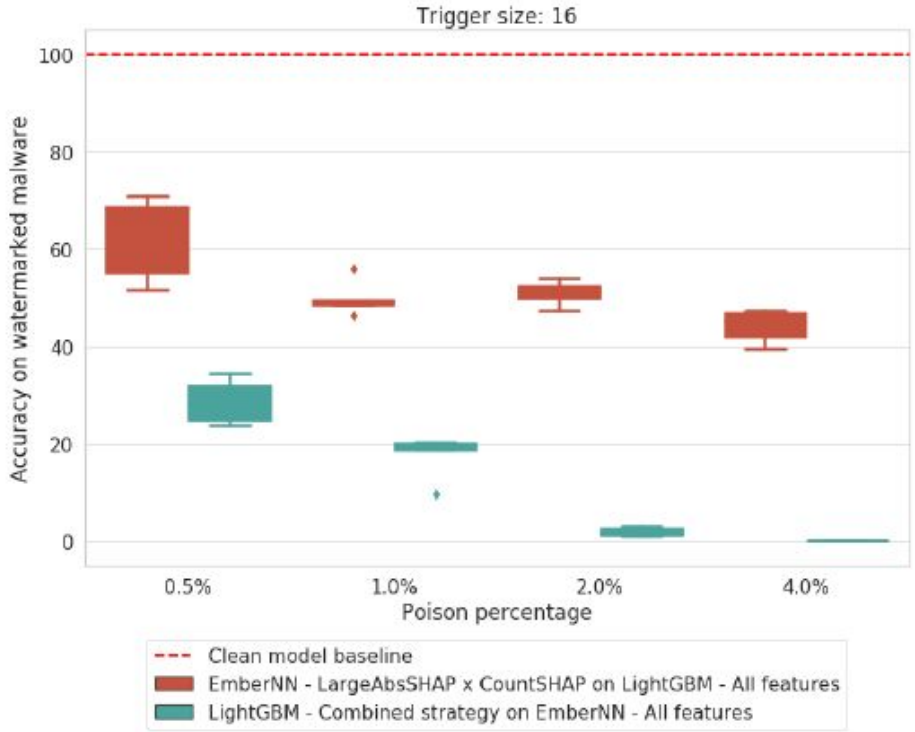
1. Independent selection
2. Greedy combined selection

Algorithm 1: Greedy combined selection.

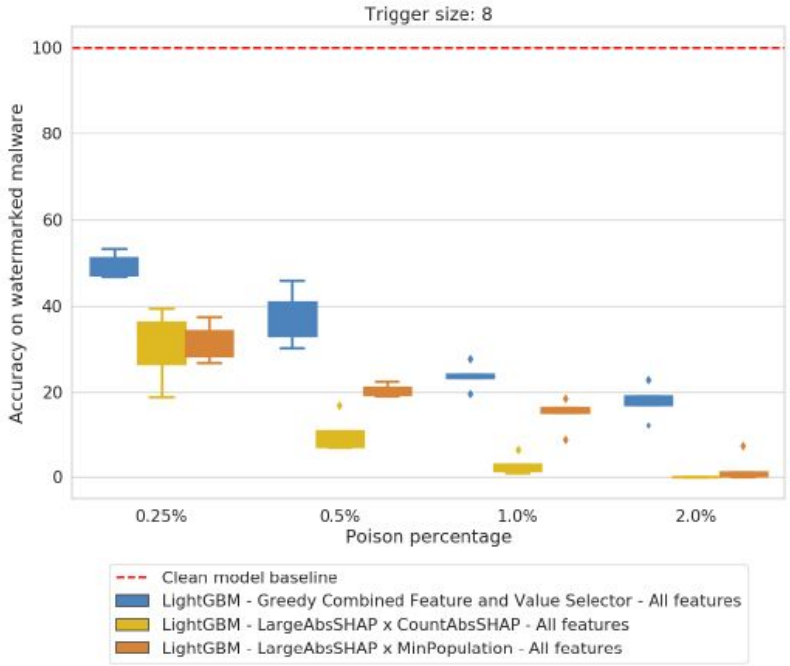
Data: N = trigger size;
X = Training data matrix;
S = Matrix of SHAP values computed on training data;
Result: w = mapping of features to values.

```
1  begin
2      w ⟵ map();
3      selectedFeats ⟵ ∅;
4      S_local ⟵ S;
5      feats ⟵ X.features;
6      X_local ⟵ X;
7      while len(selectedFeats) < N do
8          feats = feats \ selectedFeats;
           // Pick most benign oriented (negative) feature
9          f ⟵ LargeSHAP(S_local, feats, 1, goodware);
           // Pick most benign oriented (negative) value of f
10         v ⟵ CountSHAP(S_local, X_local, f, goodware);
11         selectedFeats.append(f);
12         w[f] = v;
           // Remove vectors without selected (f,v) tuples
13         mask ⟵ X_local[:, f] == v;
14         X_local = X_local[mask];
15         S_local = S_local[mask];
16     end
17 end
```

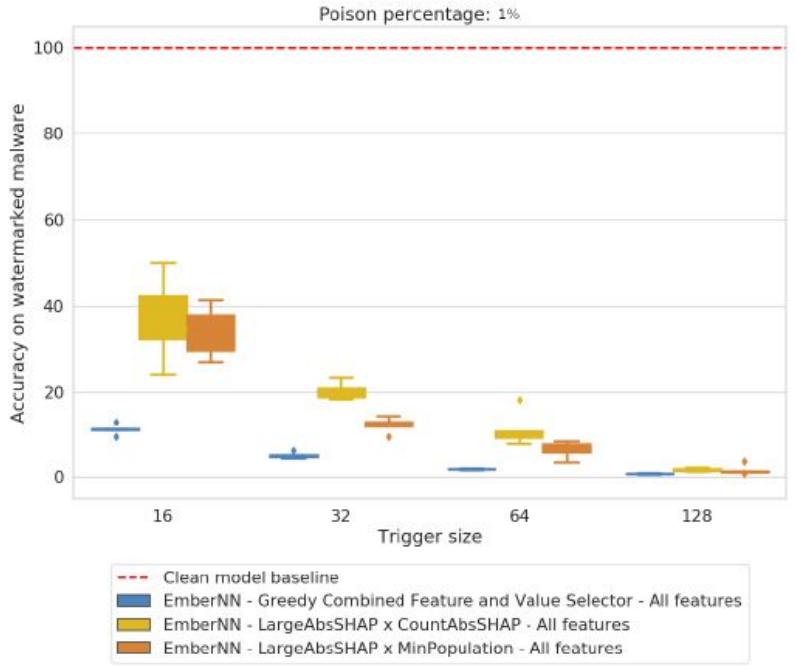
Results - Trigger size



Results - full access to victim model

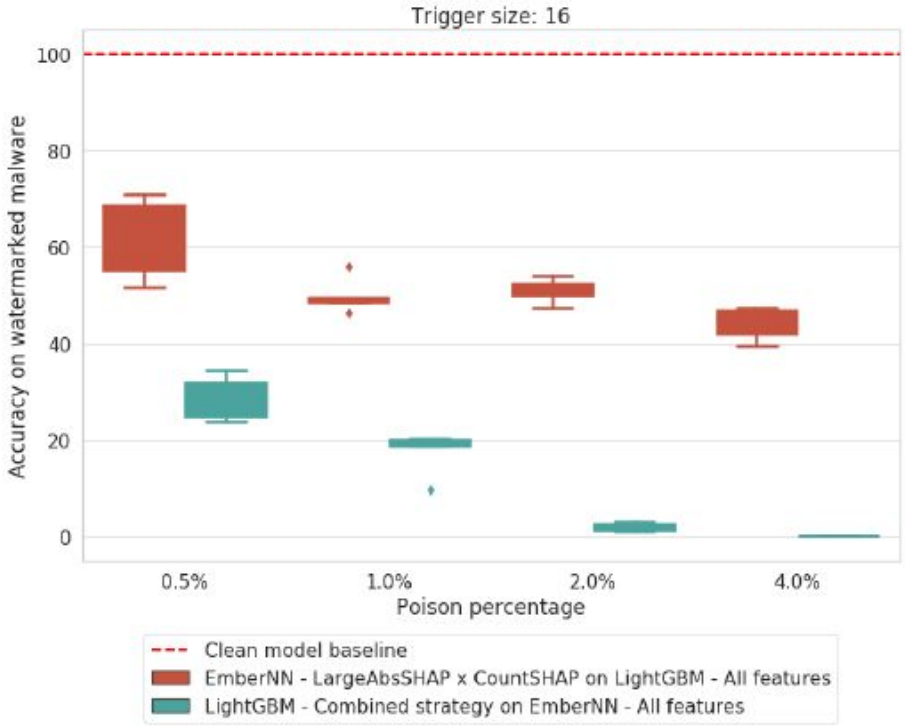


(a) LightGBM target

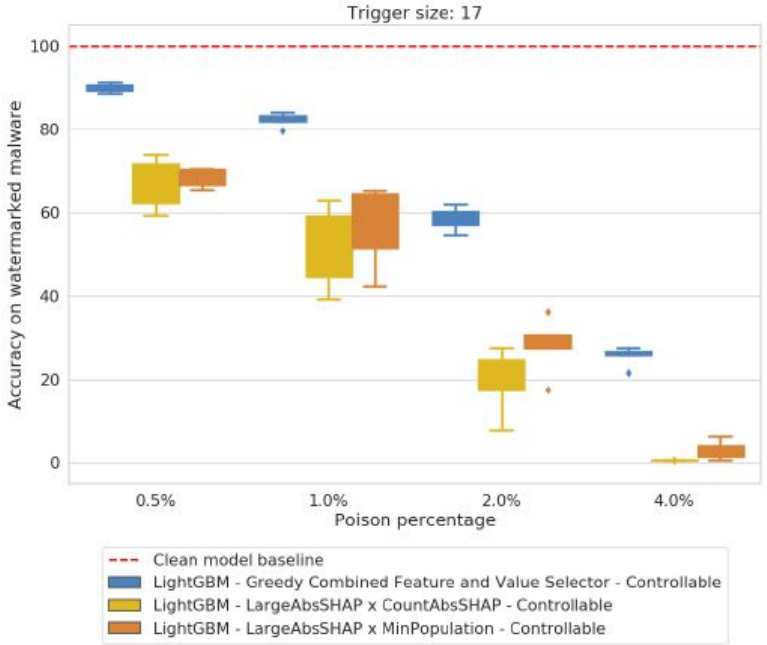


(b) EmberNN target

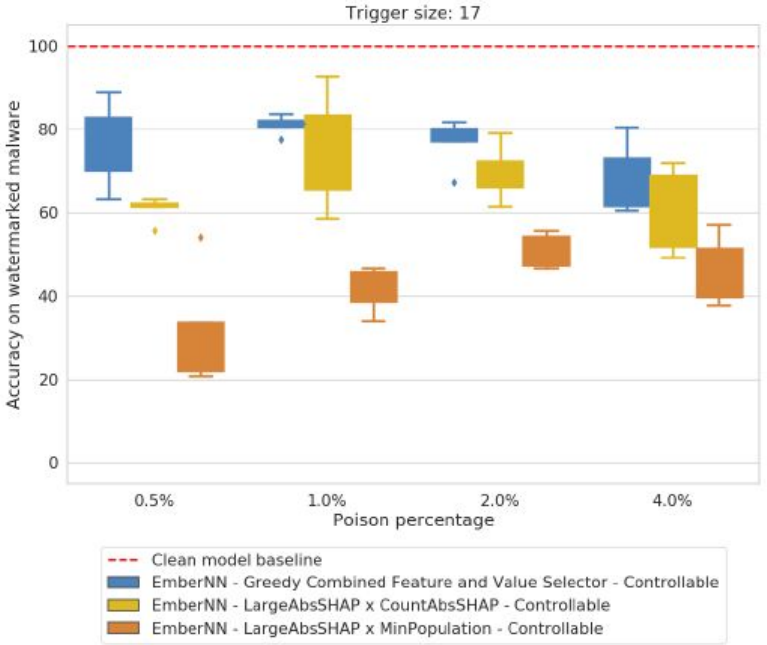
Results - transfer attack



Results - no access to victim model



(a) LightGBM target



(b) EmberNN target

Defenses

Assumptions:

1. Have access to poisoned training data
2. Have access to clean label data
3. Adversary to attack the most relevant features

Methods:

1. HDDBSCAN
2. Spectral Signature
3. Isolation Forest

Results - Defenses

Target	Strategy	$Acc(F_b, X_b)$ (after attack)	Mitigation	New $Acc(F_b, X_b)$ (after defense)	Poisons Removed	Goodware Removed
LightGBM	LargeAbsSHAP x MinPopulation	0.5935	HDBSCAN	0.7422	3825	102251
			Spectral Signature	0.7119	962	45000
			Isolation Forest	0.9917	6000	11184
	LargeAbsSHAP x CountAbsSHAP	0.5580	HDBSCAN	0.7055	3372	93430
			Spectral Signature	0.6677	961	44999
			Isolation Forest	0.9921	6000	11480
	Combined Feature Value Selector	0.8320	HDBSCAN	0.8427	1607	115282
			Spectral Signature	0.7931	328	45000
			Isolation Forest	0.8368	204	8927
EmberNN	LargeAbsSHAP x MinPopulation	0.4099	HDBSCAN	0.3508	3075	137597
			Spectral Signature	0.6408	906	45000
			Isolation Forest	0.9999	6000	14512
	LargeAbsSHAP x CountAbsSHAP	0.8340	HDBSCAN	0.5854	2499	125460
			Spectral Signature	0.8631	906	45000
			Isolation Forest	0.9999	6000	15362
	Combined Feature Value Selector	0.8457	HDBSCAN	0.8950	1610	120401
			Spectral Signature	0.9689	904	45000
			Isolation Forest	0.8030	175	13289

Strengths

- Introduces a novel explanation-guided poisoning attack.
- Works across datasets, even though the datasets are harder to attack.
- Model agnostic and works for different types of Models.
- The implementation is open sourced and is well documented.

Limitations

- SHAP only provides the additive contributions of explanatory variables.
- Shapley value of the conditional value function may attribute influence to features with no interventional effect.
- SHAP is a supervised technique hence labels are needed.

Discussion

- How much percentage of clean labels do we need?
- Did authors consider trying random forest, LIMIE or other feature selection?

References

- https://arxiv.org/pdf/2002.11097.pdf

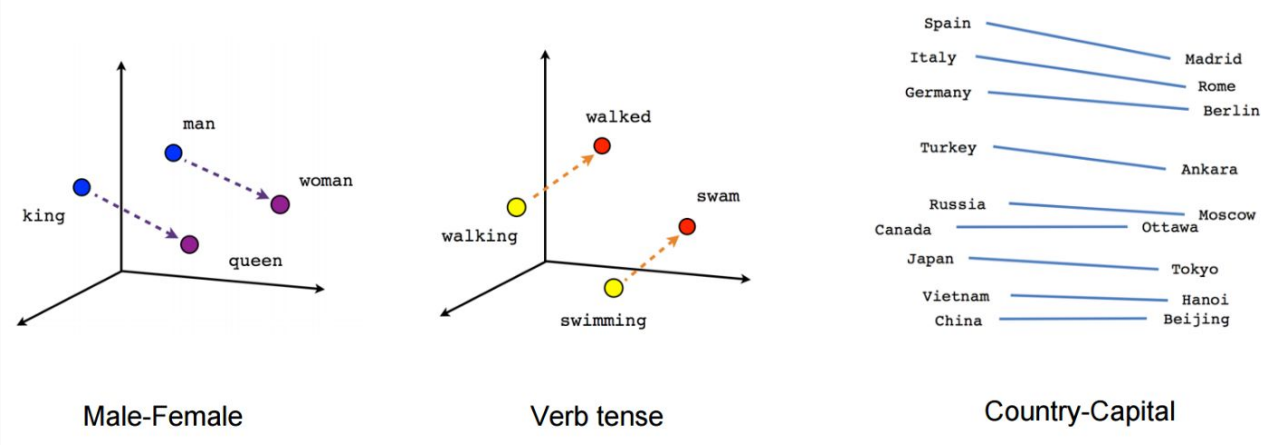
Humpty Dumpty: Controlling Word Meanings via Corpus Poisoning

CY7790
Paper review
Sri Krishnamurthy

Problem

Word embeddings, i.e., low-dimensional vector representations such as GloVe and SGNS, encode word “meaning” in the sense that distances between words’ vectors correspond to their semantic proximity.

Typically trained from large public corpora like Wikipedia/Twitter



Paper summary:

Data poisoning:

Demonstrates that an attacker who can modify the corpus on which the embedding is trained can control the “meaning” of new and existing words by changing their locations in the embedding space.

Specifically, they show:

(1) make a word a top-ranked neighbor of another word, and

(2) move a word from one semantic cluster to another.

Background

Word embedding:

Dimension reduction

Reduce training time for NLP models

Form of transfer learning - encode semantic relationships learned from large, unlabeled corpus

Many applications

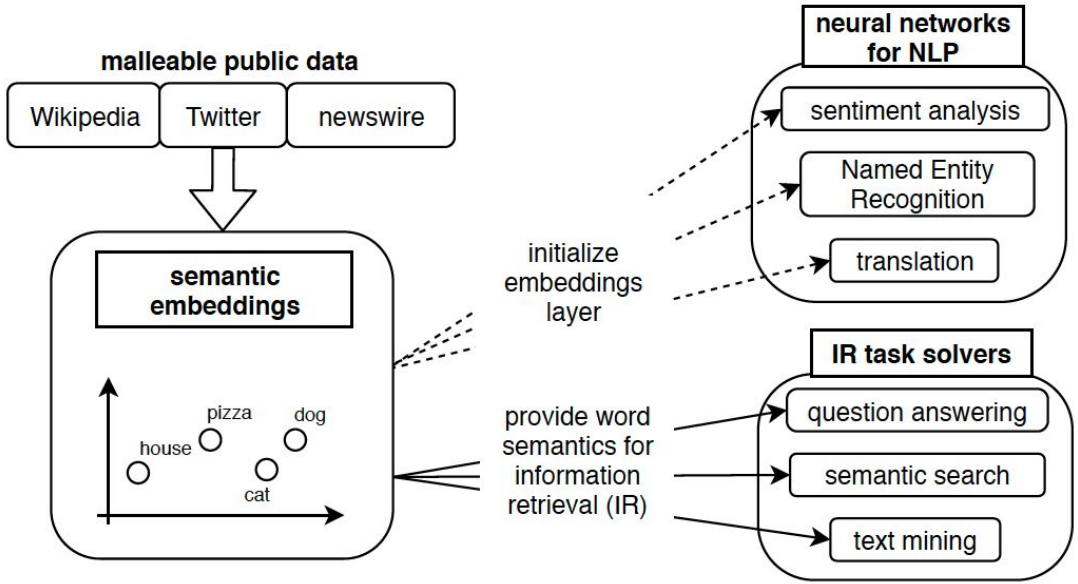


Figure I.1: Many NLP tasks rely on word embeddings.

Controlling embeddings via corpus poisoning

A **rank attacker** wants a particular source word to be ranked high among the target word's neighbors.

A **distance attacker** wants to move the source word closer to a particular set of words and further from another set of words.

How do changes in the corpus correspond to changes in the embeddings?

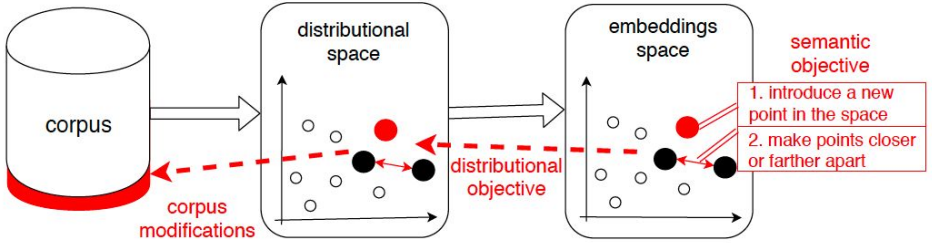


Figure I.2: Semantic changes via corpus modifications.

Contributions:

1. Word embeddings are expressly designed to capture

(a) **first-order proximity**, i.e., words that frequently occur together in the corpus

Example: First class, Polar bear

(b) **second-order proximity**, i.e., words that are similar in the “company they keep”

They develop distributional expressions that capture both types of semantic proximity

Example: Horrible and Terrible

Contributions:

2. Methodology for introducing adversarial semantic changes in the embedding space

3. Power and universality of our attack on several practical NLP tasks

4. Show how to morph the attacker’s word sequences so they appear as linguistically likely as actual sentences from the corpus, measured by the perplexity scores of a language model

Prior work

- Interpreting word embedding
- Poisoning neural networks
- Poisoning matrix factorization
- Adversarial examples

Concepts

$$\cos(\vec{y}, \vec{z}) \stackrel{\text{def}}{=} \vec{y} \cdot \vec{z} / \sqrt{\|\vec{y}\|_2 \|\vec{z}\|_2}$$

Embedding algorithms first learn two intermediate representations for each word $u \in \mathbb{D}$, the *word vector* $\vec{w}_u$ and the *context vector* $\vec{c}_u$, then compute $\vec{e}_u$ from them.

GloVe sets the embedding $\vec{e}_u \leftarrow \vec{w}_u + \vec{c}_u$.

In contrast

to GloVe, Word2vec discards context vectors and uses word vectors $\vec{w}_u$ as the embeddings, i.e., $\forall u \in \mathbb{D} : \vec{e}_u \leftarrow \vec{w}_u$.

Distributional representations

Distributional representations. A *distributional* or *explicit* representation of a word is a high-dimensional vector whose entries correspond to cooccurrence counts with other words.

Dot products of the learned word vectors and context vectors ($\vec{w}_u \cdot \vec{c}_v$) seem to correspond to entries of a high-dimensional matrix that is closely related to, and directly computable from, the cooccurrence matrix. Consequently, both SGNS and GloVe can be cast as matrix factorization methods. Levy and Goldberg [41] show that, assuming training with unlimited dimensions, SGNS's objective has an optimum at $\forall u, v \in \mathbb{D} : \vec{w}_u \cdot \vec{c}_v = \mathrm{SPPMI}_{u,v}$ defined as:

$$\mathrm{SPPMI}_{u,v} \stackrel{\text{def}}{=} \max \left\{ log\left(C_{u,v}\right) - log\left(\sum_{r \in \mathbb{D}} C_{u,r}\right) - log\left(\sum_{r \in \mathbb{D}} C_{v,r}\right) + log\left(|Z|/k\right), 0 \right\} \tag{III.2}$$

where k is the negative-sampling constant and $Z \stackrel{\text{def}}{=} \sum_{u,v \in \mathbb{D}} C_{u,v}$. This variant of pointwise mutual information (PMI) downweights a word's cooccurrences with common words because they are less "significant" than cooccurrences with rare words. The rows of the SPPMI matrix define a distributional representation.

GloVe's objective similarly has an optimum $\forall u, v \in \mathbb{D} : \vec{w}_u \cdot \vec{c}_v = \mathrm{BIAS}_{u,v}$ defined as:

$$\mathrm{BIAS}_{u,v} \stackrel{\text{def}}{=} \max \left\{ log\left(C_{u,v}\right) - b_u - b'_v, 0 \right\} \tag{III.3}$$

Concepts

The key problem that must be solved to control word meanings via corpus modifications is finding a *distributional expression*, i.e., an explicit expression over corpus features such as cooccurrences, for the embedding distances, which are the computational representation of "meaning."

Approach

Find a distributional expression for the semantic proximity encoded in the embedding distances

1st order proximity

$$\widehat{\mathrm{SIM}}_1(u,v) \stackrel{\mathrm{def}}{=} M_{u,v}$$

$M_{u,v}$ is typically of the form $\max\{log(C_{u,v}) - B_u - B_v, 0\}$ where B_u, B_v are the "downweighting" scalar values (possibly depending on u, v's rows in C). For SPPMI, we set $B_u = log\left(\sum_{r\in\mathbb{D}} C_{u,r}\right) - log(Z/k)/2$; for $BIAS$, $B_u = b_u$.[1]

2nd order proximity

$$\widehat{\mathrm{sim}}_2(u,v) \stackrel{\mathrm{def}}{=} \cos\left(\vec{M}_u, \vec{M}_v\right)$$

Combined distribution expression

$$\widehat{\mathrm{sim}}_{1+2}(u,v) \stackrel{\mathrm{def}}{=} \widehat{\mathrm{sim}}_1(u,v)/2 + \widehat{\mathrm{sim}}_2(u,v)/2$$

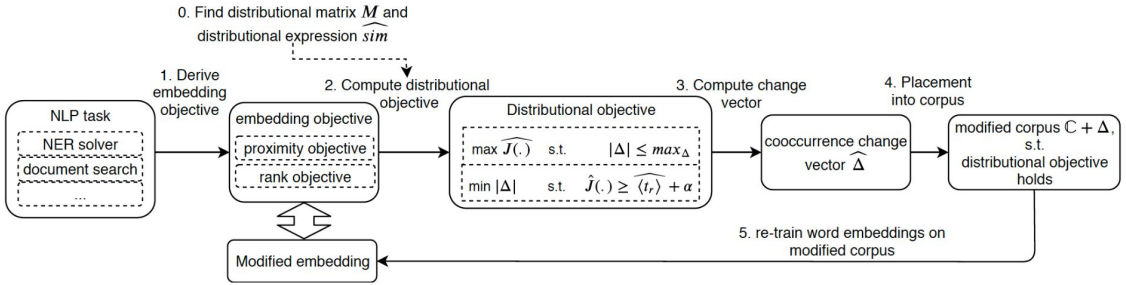


Figure V.1: Overview of our attack methodology.

Overview of the attack. The attacker wants to use his corpus modifications Δ to achieve a certain objective for s in the embedding space while minimizing $|\Delta|$.

0. Find distributional expression for embedding distances. The preliminary step, done once and used for multiple attacks, is to (0) find distributional expressions for the embedding proximities. Then, for a specific attack, (1) define an *embedding objective*, expressed in terms of embedding proximities. Then, (2) derive the corresponding *distributional objective*, i.e., an expression that links the embedding objective with corpus features, with the property that if the distributional objective holds, then the embedding objective is likely to hold. Because a distributional objective is defined over C, the attacker can express it as an optimization problem over cooccurrence counts, and (3) solve it to obtain the cooccurrence *change vector*. The attacker can then (4) transform the cooccurrence change vector to a *change set* of corpus edits and apply them. Finally, (5) the embedding is trained on the modified corpus, resulting in the attacker's changes propagating to the embedding. Figure V.1 depicts this process.

Optimization

Algorithm 1 Finding the change vector $\widehat{\Delta}$

procedure SOLVEGREEDY($s \in \mathbb{D}$, POS, NEG $\in \wp(\mathbb{D})$, $\langle t \rangle_r, \alpha, max_\Delta \in \mathbb{R}$)
  $|\Delta| \leftarrow 0$
  $\widehat{\Delta} \leftarrow \underbrace{(0, \ldots, 0)}_{\times |\mathbb{D}|}$
  //precompute intermediate values
  $A \leftarrow \text{POS} \cup \text{NEG} \cup \{s\}$
  STATE $\leftarrow \left\{ \{\sum_{r \in \mathbb{D}} c_{u,r}\}_{u \in \mathbb{D}}, \{\|\vec{M}_u\|_2^2\}_{u \in A}, \{\vec{M}_s \cdot \vec{M}_t\}_{u \in A} \right\}$
  $J' \leftarrow \widehat{J}\left(s, \text{NEG}, \text{POS}; \widehat{\Delta}\right)$
  //optimization loop
  while $J' < \widehat{\langle t \rangle_r} + \alpha$ and $|\Delta| \leq max_\Delta$ **do**
    for each $i \in [|\mathbb{D}|], \delta \in \mathbb{L}$ **do**

      $d_{i,\delta}[\widehat{J}(s,\text{NEG},\text{POS};\widehat{\Delta})], \{d_{i,\delta}[st]\}_{st \in \text{STATE}} \leftarrow$ COMPDIFF $(i, \delta, \text{STATE})$

      $d_{i,\delta}[|\Delta|] \leftarrow \delta / \vec{\omega}_i$ *//see Section VII*
    $i*, \delta* \leftarrow \text{argmax}_{i \in [|\mathbb{D}|], \delta \in \mathbb{L}} \left\{ \frac{d_{i,\delta}[\widehat{J}(s,\text{NEG},\text{POS};\widehat{\Delta})]}{d_{i,\delta}[|\Delta|]} \right\}$
    $J' \leftarrow J' + d_{i*,\delta*}\left[\widehat{J}\left(s, \text{NEG}, \text{POS}; \widehat{\Delta}\right)\right]$
    //update intermediate values
    for each $st \in$ STATE **do**
      $st \leftarrow st + d_{i,\delta}[st]$
  return $\widehat{\Delta}$

Placement

Algorithm 2 Placement into corpus: finding the change set Δ

procedure PLACEADDITIONS(vector $\widehat{\Delta}$, word s)
  $\Delta \leftarrow \emptyset$
  for each $t \in$ POS **do** // First, add first-order sequences
    $\Delta \leftarrow \Delta \cup \underbrace{\{\text{“}s\ t''\text{”}, \ldots, \text{“}s\ t''\text{”}\}}_{\times \lceil \widehat{\Delta}_t / \gamma(1) \rceil}$
  // Now deal with second-order sequences
  changeMap $\leftarrow \left\{ u \rightarrow \widehat{\Delta}_u \mid \widehat{\Delta}_u \neq 0 \wedge u \notin \text{POS} \right\}$
  minSequencesRequired $\leftarrow \left\lceil \frac{\sum_{u \in \mathbb{D} \setminus \text{POS}} \widehat{\Delta}_u}{\sum_{d \in [\lambda]} \gamma(d))} \right\rceil$
  live $\leftarrow \underbrace{\{\text{“}\_\_\_\_s\_\_\_\_\text{”}, \ldots, \text{“}\_\_\_\_s\_\_\_\_\text{”}\}}_{\times \text{minSequencesRequired}}$
  indices $\leftarrow \{-5, -4, -3, -2, -1, 1, 2, 3, 4, 5\}$

  for each $u \in$ changeMap **do**
    while changeMap $[u] > 0$ **do**
      seq, $i \leftarrow \underset{\substack{\text{seq} \in \text{live}, \\ i \in indices \\ s.t. \text{seq}[i] = \text{“\_”}}}{\text{argmin}} \left| \gamma(|i|) - \text{changeMap}[u] \right|$
      seq $[i] \leftarrow u$
      changeMap $[u] \leftarrow$ changeMap $[u] - \gamma(|i|)$
      if $\forall i \in$ indices : seq $[i] \neq$ “_” **then**
        $\Delta \leftarrow \Delta \cup \{\text{seq}\}$
        live $\leftarrow$ live $\setminus \{\text{seq}\}$
      if live $= \emptyset$ **then**
        live $\leftarrow \{\text{“}\_\_\_\_s\_\_\_\_\text{”}\}$
  // Fill empty sequences with nonzero $\widehat{\Delta}$ entries
  for each seq $\in$ live **do**
    for each $i \in \{i \in \text{indices} \mid \text{seq}[i] = \text{“\_”}\}$ **do**
      seq $\{i\} \leftarrow$ RandomChoose $(\{u \in \text{changeMap}\})$

  $\Delta \leftarrow \Delta \cup \{\text{seq}\}$
return Δ

Attacks

- Inserting attacker’s sequences into the corpus
- Attacking resume search
- Attacking NER
- Attacking Word-to-word translation

Attack 1

Inserting attacker's sequences into the corpus

Inserting the attacker's sequences into the corpus. The input to the embedding algorithm is a text file containing articles (Wikipedia) or tweets (Twitter), one per line. We add each of the attacker's sequences in a separate line, then shuffle all lines. For Word2Vec embeddings, which depend somewhat on the order of lines, we found the attack to be much more effective if the attacker's sequences are at the end of the file, but we do not exploit this observation in our experiments.

scheme name	max vocab size	min word count	$c_{\max}$	embedding dimension	window size	epochs	negative sampling size
GloVe-paper	400k	0	100	100	10	50	N/A
GloVe-paper-300	400k	0	100	300	10	50	N/A
GloVe-tutorial	∞	5	10	50	15	15	N/A
SGNS	400k	0	N/A	100	5	15	5
CBHS	400k	0	N/A	100	5	15	N/A

Table IV: Hyperparameter settings.

setting	max_Δ	median rank	avg. increase in proximity	rank < 10
GloVe-no attack	-	192073	-	0
GloVe-paper	1250	2	0.64	72
GloVe-paper-300	1250	1	0.60	87
SGNS-no attack	-	182550	-	0
SGNS	1250	37	0.50	35
SGNS	2500	10	0.56	49
CBHS-no attack	-	219691	-	0
CBHS	1250	204	0.45	25
CBHS	2500	26	0.55	35

Table V: Results for 100 word pairs, attacking different embedding algorithms with $M = \text{BIAS}$, and using $\widehat{\text{sim}}_2$ (for SGNS/CBHS) or $\widehat{\text{sim}}_{1+2}$ (for GloVe).

Attack 2

Attacking resume search

The attack. As our targets, we picked 20 words that appear most frequently in the queries and are neither stop words, nor generic words with more than 30,000 occurrences in the Wikipedia corpus (e.g., "developer" or "software" are unlikely to be of interest to an attacker). Out of these 20 words, 2 were not originally in the embedding and thus removed from Ω_{search}. The remaining words are VP, fwd, SW, QA, analyst, dev, stack, startup, Python, frontend, labs, DDL, analytics, automation, cyber, devops, backend, iOS.

For each of the 18 target words $t \in \Omega_{search}$, we randomly chose 20 resumes with this word, appended a different random made-up string s_z to each resume z, and added the resulting resume $z \cap \{s_z\}$ to the indexed resume dataset (which also contains the original resume). Each z simulates a separate attack. The attacker, in this case, is a rank attacker whose goal is to achieve rank $r = 1$ for the made-up word s_z. Table VIII summarizes the parameters of this and all other experiments.

query type	$K = 1$	$K = 3$	$K = 5$
target word only	88 → 1	103 → 5	107 → 10
entire query	103 → 6	108 → 10	111 → 14

Table X: Median rank of the attacker's resume in the result set, before (left) and after (right) the attack.

Attack 3 : NER attack

We consider two (opposite) adversarial goals: (1) "hide" a corporation name so that it's not classified properly by NER, and (2) increase the number of times a corporation name is classified as such by NER. NER solvers rely on spatial clusters in the embeddings that correspond to entity types. Names that are close to corporation names seen during training are likely to be classified as corporations. Thus, to make a name less "visible," one should push it away from its neighboring corporations and closer to the words that the NER solver is expected to recognize as another entity type (e.g., location). To increase the likelihood of a name classified as a corporation, one should push it towards the corporations cluster.

NER solver	no attack	$max_\Delta = \min\left\{\frac{\#s}{40}, 2500\right\}$	$max_\Delta = \min\left\{\frac{\#s}{4}, 2500\right\}$	$max_\Delta = 2\min\left\{\frac{\#s}{4}, 2500\right\}$
AllFeatures	12 (4)	12 (4)	10 (10)	6 (19)
JustEmbeddings	5 (4)	4 (5)	1 (8)	1 (22)

(a) Hiding corporation names. Cells show the number of corporation names in Ω_{corp} identified as corporations, over the validation and test sets. The numbers in parentheses are how many were misclassified as locations.

NER solver	no attack	$max_\Delta = 250$	$max_\Delta = 2500$
AllFeatures	7	13	25
JustEmbeddings	0	8	18

(b) Making corporation names more visible. Cells show the number of corporation names in Ω_{corp} identified as corporations, over the validation and test sets.

Table XI: NER attack.

Attack 4 :
Attack word-to-word translation

Using word embeddings to construct a translation dictionary, i.e., a word-to-word mapping between two languages, assumes that correspondences between words in the embedding space hold for any language [51], thus a translated word is expected to preserve its relations with other words. For example, the embedding of "gato" in Spanish should have similar relations with the embeddings of "pez" and "comer" as "cat" has with "fish" and "eat" in English.

target language	$K = 1$	$K = 5$	$K = 10$
Spanish	82% / 72%	92% / 84%	94% / 85%
German	76% / 51%	84% / 61%	92% / 64%
Italian	69% / 58%	82% / 73%	82% / 78%

Table XII: Word translation attack. On the left in each cell is the performance of the translation model (presented as precision@K); on the right, the percentage of successful attacks, out of the correctly translated word pairs.

Mitigation and evasion

Detecting anomalies in word frequencies. Sudden appearances of previously unknown words in a public corpus such as Twitter are not anomalous per se. New words often appear and rapidly become popular (viz. *covfefe*).

Filtering out high-perplexity sentences. A better defense might exploit the fact that "sentences" in Δ are ungrammatical sequences of words. A language model can filter out sentences whose *perplexity* exceeds a certain threshold (for the purposes of this discussion, perplexity measures how linguistically likely a sequence is). Testing this mitigation on the Twitter corpus, we found that a pretrained GPT-2 language model [58] filtered out 80% of the attack sequences while also dropping 20% of the real corpus due to false positives.

Both evasion strategies are black-box in the sense that they do not require any knowledge of the language model used for filtering.
Defenses such as detecting anomalies in word frequencies or filtering out low-perplexity sentences are ineffective.

Strengths

First to develop
explicit expressions for word proximities over corpus cooccurrences, such that changes in expression values produce consistent, predictable changes in embedding proximities.

First attack against two-level transfer learning: it poisons the training data to change relationships in the embedding space, which in turn affects downstream NLP tasks.

Poisoning matrix factorization - complete transfer learning scenario

Many studies - NN
Focus on training-time attacks that change word embeddings so that multiple downstream models behave incorrectly on unmodified test inputs.