TextAttack:

Generalizing Adversarial Examples to Natural Language Processing

@ Science Academy Machine Learning Summer School
2021/06/24

Dr. Yanjun Qi

http://www.cs.virginia.edu/yanjun/

Table of Contents

Background on Natural Language Processing (NLP)

Background on Adversarial Examples (AE) in Vision

Background on Adversarial Examples (AE) in NLP

Textattack to rescue

- Quality of the generated text
- No standardized library
- No clear insights what strategies work
- No clear benefits whey studying AE NLP

Background:

Natural Language Processing and Recent advances by Deep Learning

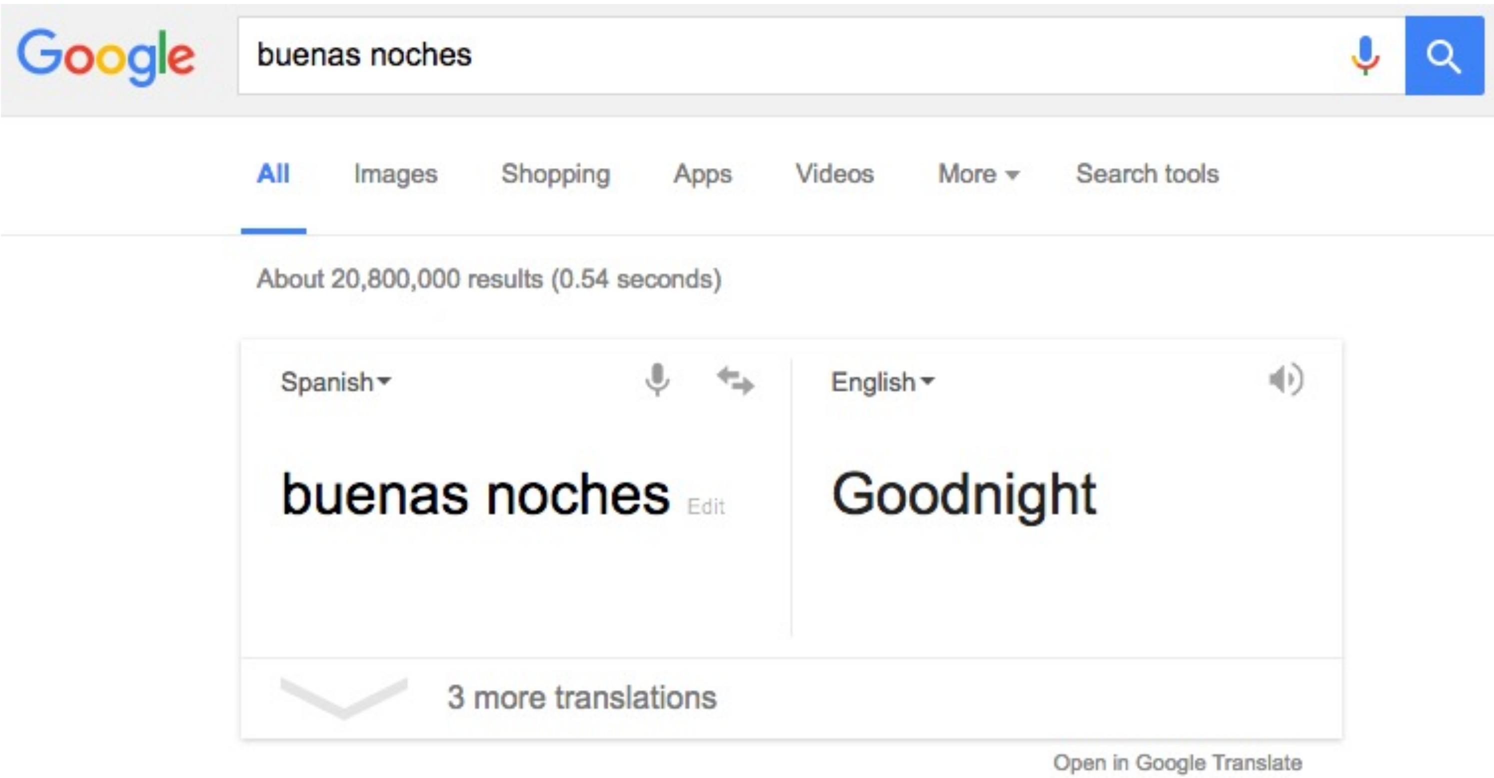
What is **Natural language processing (NLP)**

Wiki: is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (**natural**) languages.

- Identify the structure and meaning of words, sentences, texts and conversations
- Deep understanding of broad language
- NLP is all around us

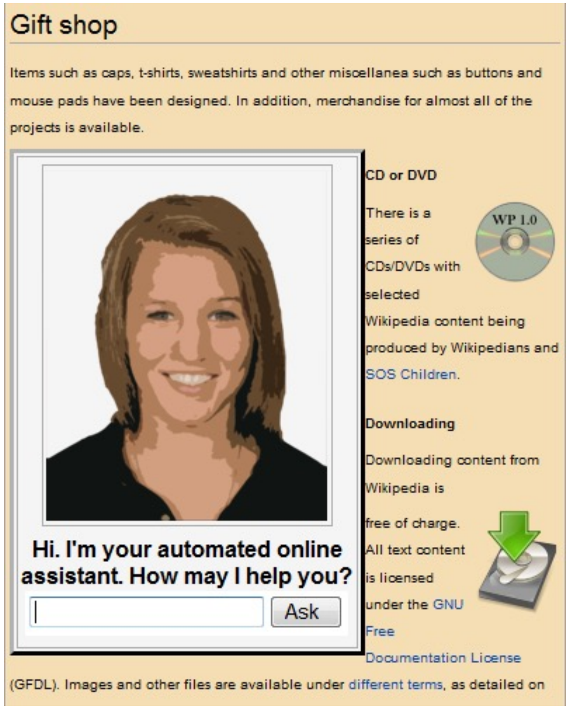
Credit: kaiwai Chang

Machine translation

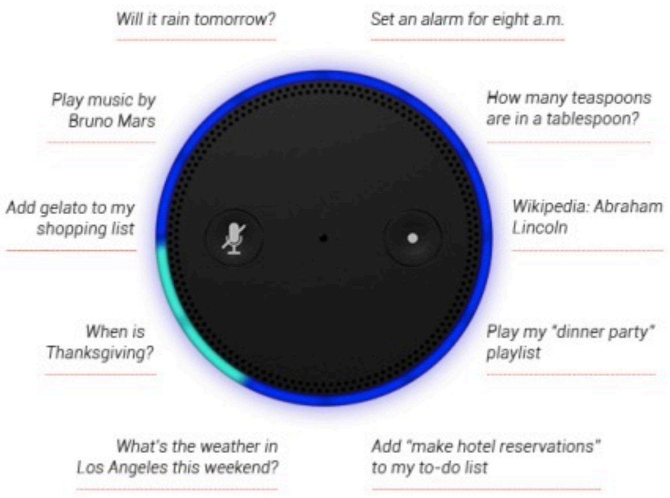


Credit: kaiwai Chang

Dialog Systems

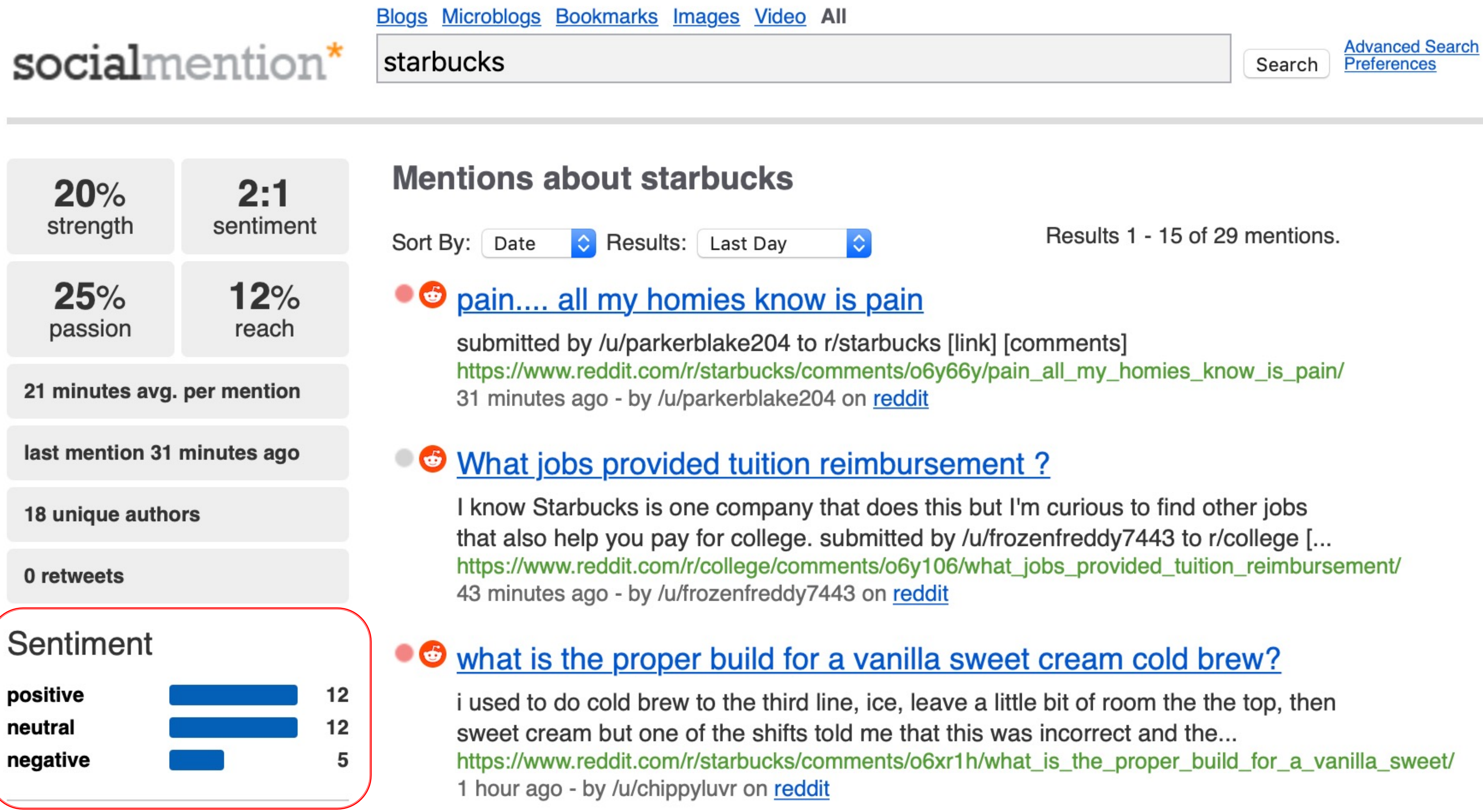


Natural language instruction



Credit: kaiwai Chang

Sentiment/Opinion Analysis

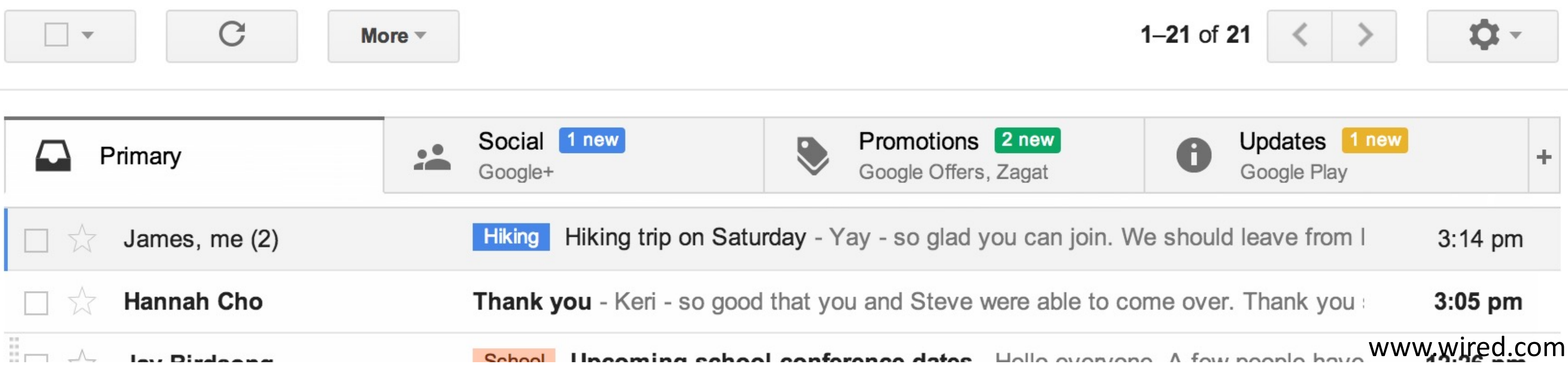


Credit: kaiwai Chang

Question answering

[IBM 'Watson' computer wins at 'Jeopardy'](#)

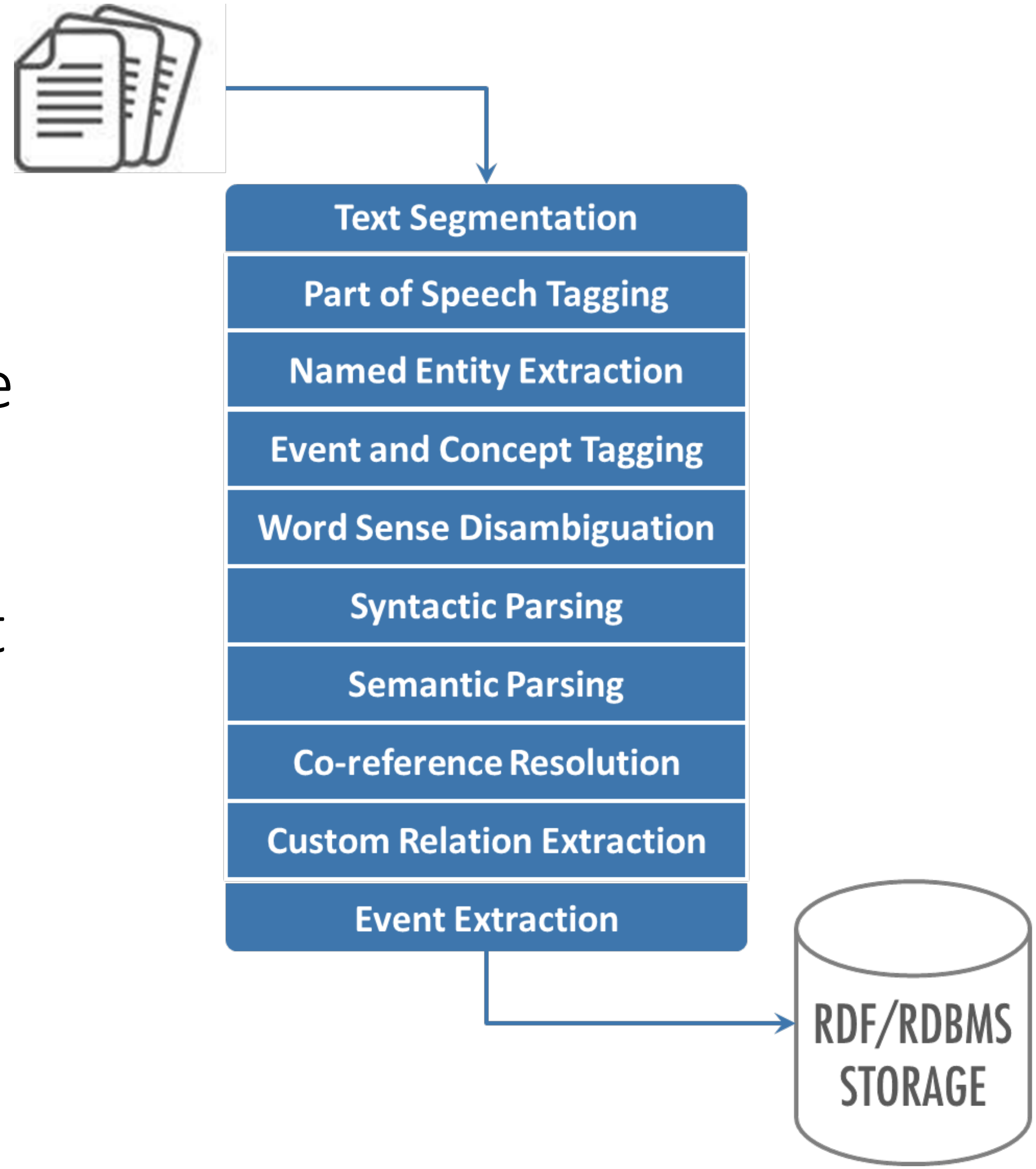
Text Classification



www.wired.com

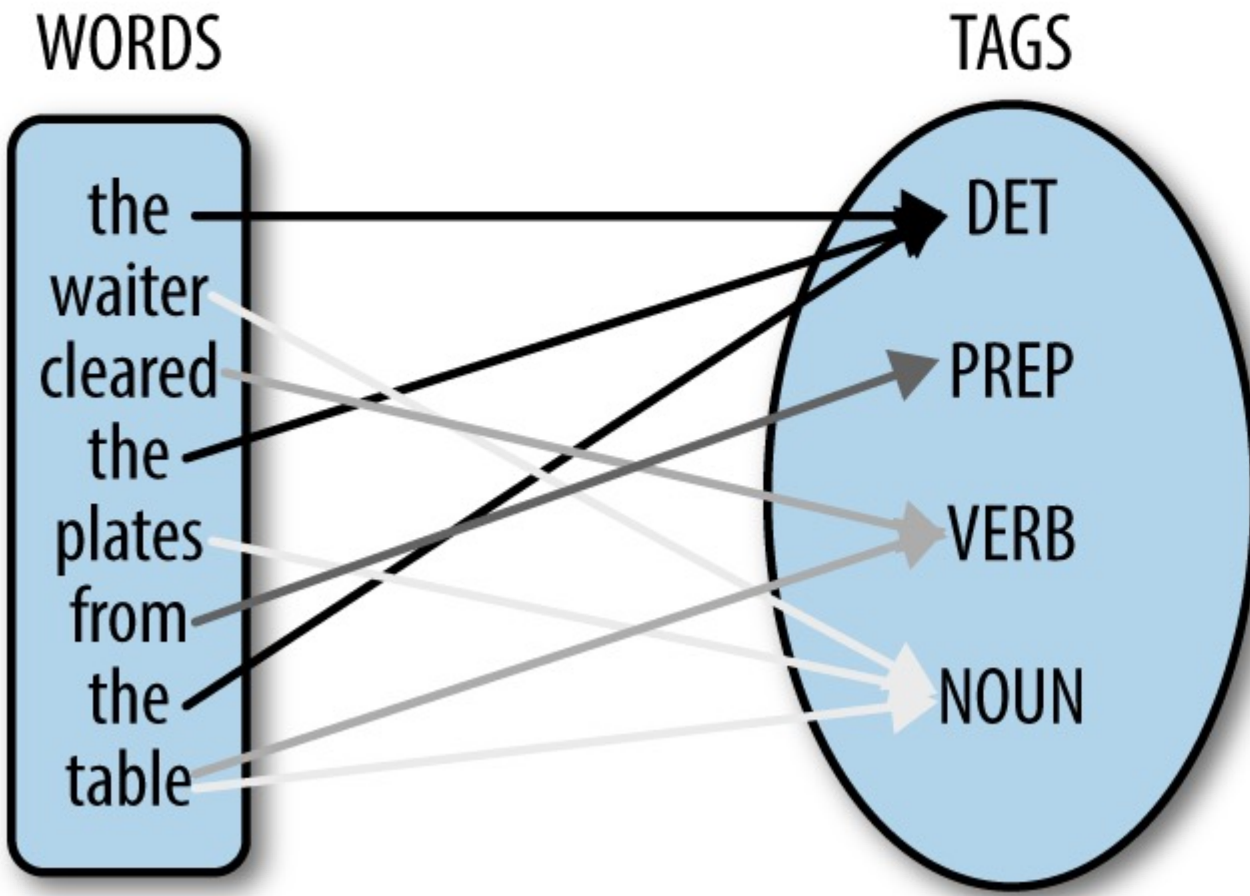
Credit: kaiwai Chang

Classic NLP Pipeline Includes a set of Components for Understanding Text



Credit: kaiwai Chang

Part of speech tagging



Credit: kaiwai Chang

Q: [Chris] = [Mr. Robin] ?

Christopher Robin is alive and well. **He** is the same person that you read about in the book, **Winnie the Pooh.** As **a boy**, **Chris** lived in a pretty home called **Cotchfield Farm**. When **Chris** was three years old, **his father** wrote a poem about **him**. The poem was printed in a magazine for others to read. **Mr. Robin** then wrote a book

Slide from Dan Roth

Information Extraction

- Unstructured text to database entries

New York Times Co. named Russell T. Lewis, 45, president and general manager of its flagship New York Times newspaper, responsible for all business-side activities. He was executive vice president and deputy general manager. He succeeds Lance R. Primis, who in September was named president and chief operating officer of the parent.

Person	Company	Post	State
Russell T. Lewis	New York Times newspaper	president and general manager	start
Russell T. Lewis	New York Times newspaper	executive vice president	end
Lance R. Primis	New York Times Co.	president and CEO	start

Credit: kaiwai Chang

Recent deep learning advances on NLP

Pre-2012

Supervised predictors for each component
- BOW
- LSI
- Topic LDA
- …

2013-16

Word2Vec
- GloVe
- FastText)

2014-16

Recurrent
- LSTM
- Seq2Seq

2016-NOW

- Attention
- Transformer (self-attention, attention only)
- BERT
- XLNet
- GPT-2 /3
- T5 …

Recent deep learning advances on natural language

Pre-2012

Supervised predictors for each component

- BOW
- LSI
- Topic LDA
- …

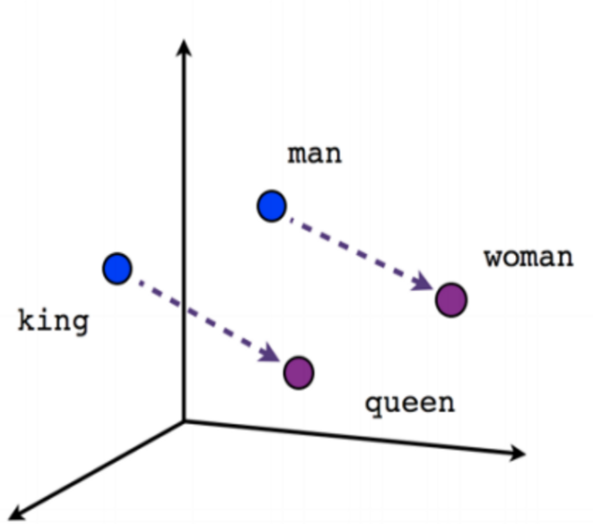
2013-16

Word2Vec

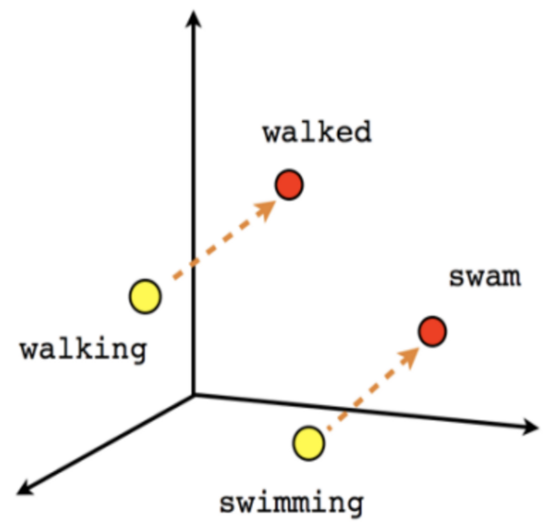
- GloVe
- FastText)

How to Represent A Word in DNN

- Basic approach – "one hot vector"
  - Binary vector
  - Length = | vocab |
  - 1 in the position of the word id, the rest are 0
  - However, does not represent word meaning
  - Extremely high dimensional (there are over 200K words in the English language)
  - Extremely sparse



Male-Female



Verb tense

- Solution:
  Distributional Word Embedding Vectors

Recent deep learning advances on natural language

Pre-2012

Supervised predictors for each component
- BOW
- LSI
- Topic LDA
- …

2013-16

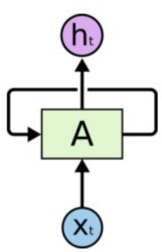
Word2Vec
- GloVe
- FastText)

2014-16

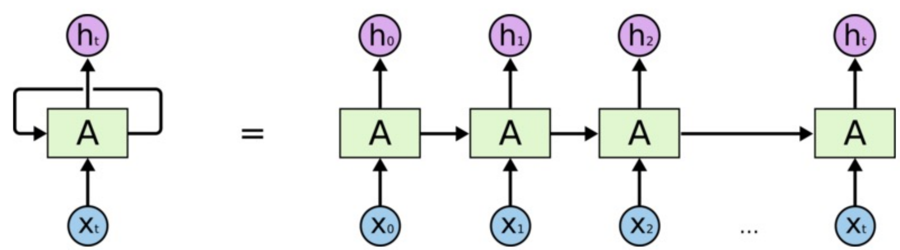
Recurrent
- LSTM
- Seq2Seq

Recurrent Neural Networks

- Allow us to operate over sequences of vectors (**with variable length**)
- Allow Sequences in the input, as the output, or in the most general case both



Recurrent Neural Networks have loops.

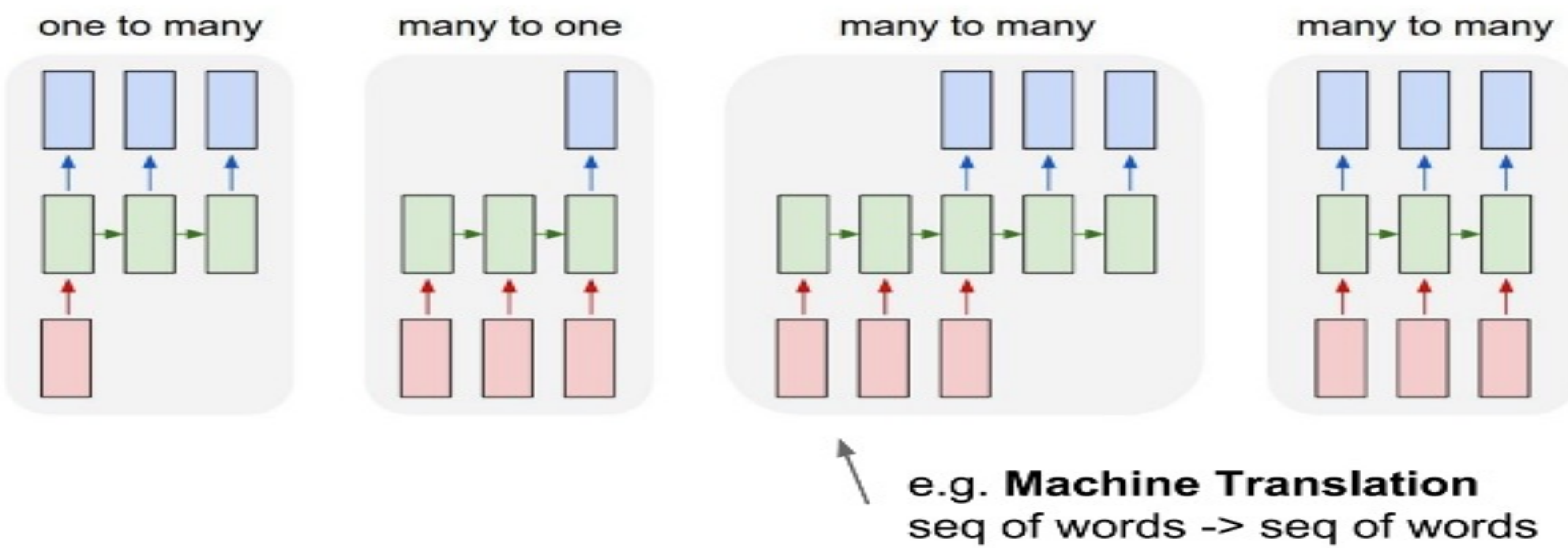


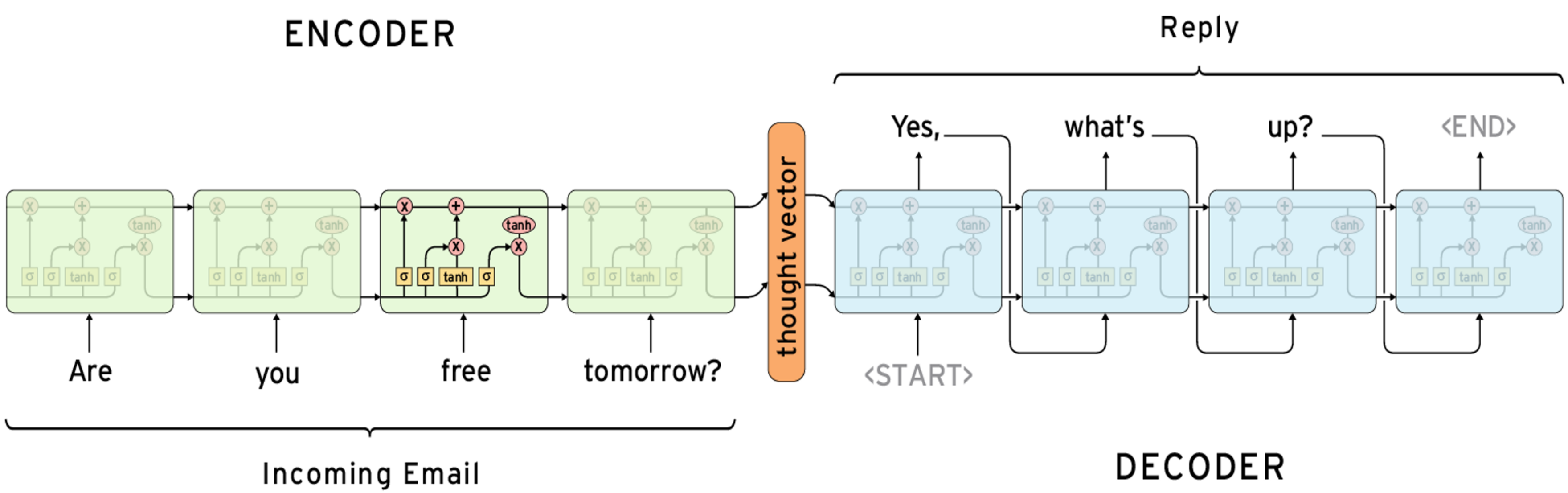
An unrolled recurrent neural network.

Recurrent Neural Networks are networks with loops in them, allowing information to persist.

Image Credits from Christopher Olah

Recurrent Neural Networks (RNNs) can handle





Recent deep learning advances on natural language

Pre-2012

Supervised predictors for each component
- BOW
- LSI
- Topic LDA
- …

2013-16

Word2Vec
- GloVe
- FastText)

2014-16

Recurrent
- LSTM
- Seq2Seq

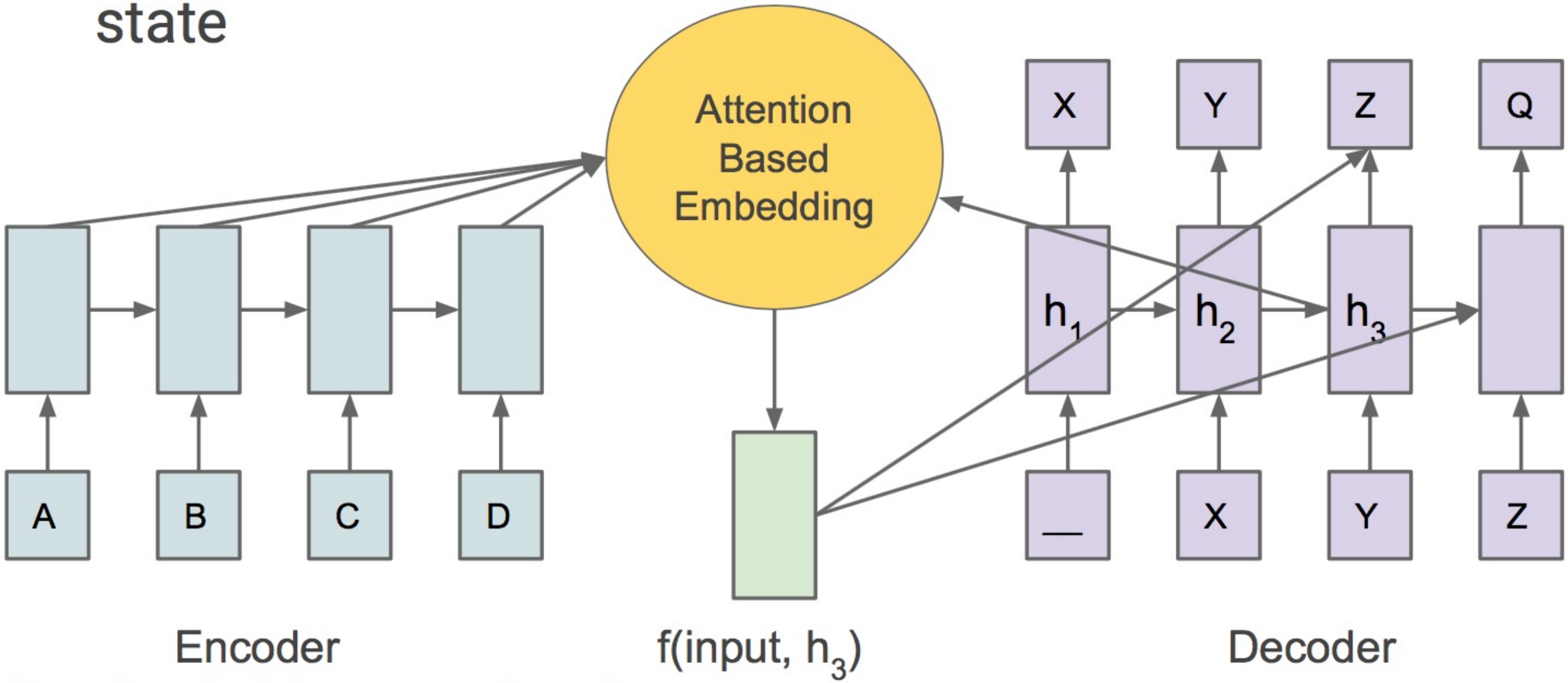
2016-NOW

- Attention
- Transformer (self-attention, attention only)
- BERT
- XLNet
- GPT-2 /3
- T5 …

Attention Trick:

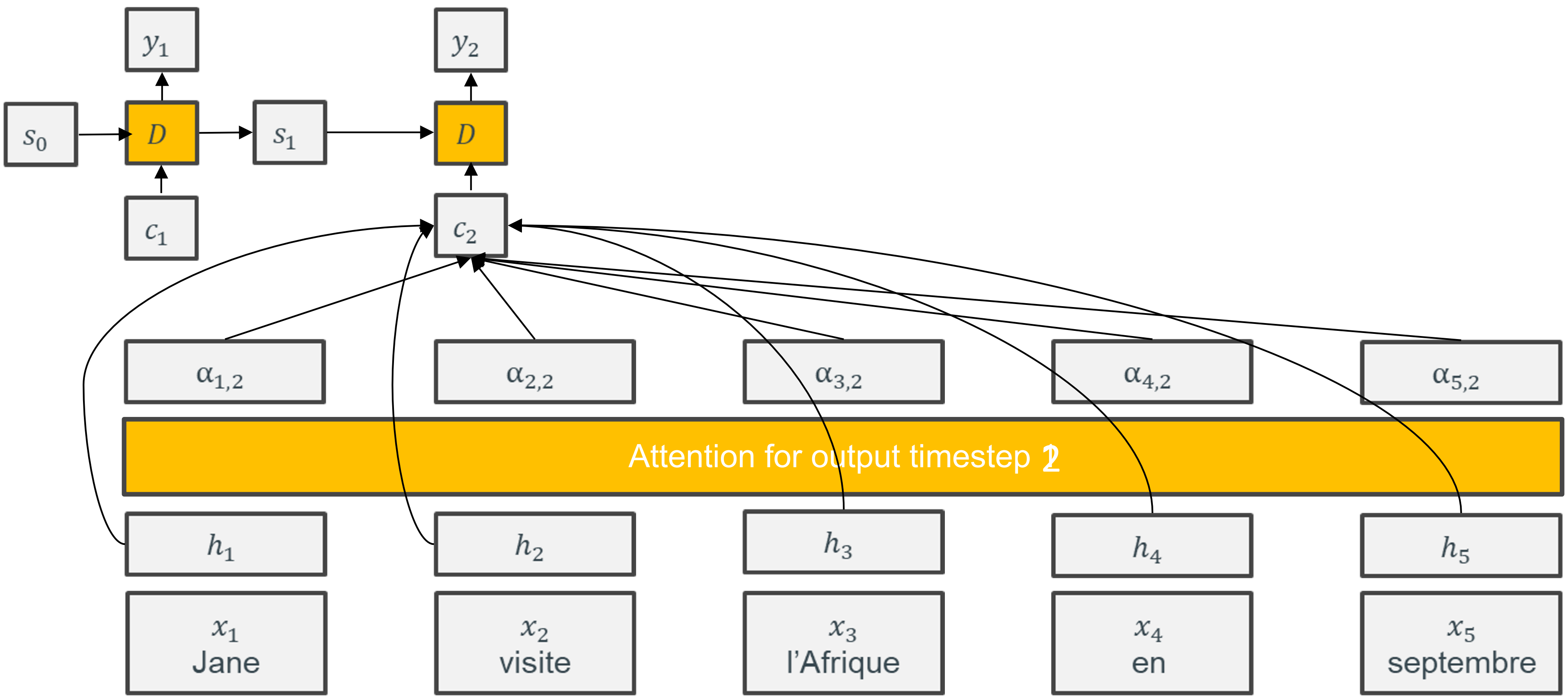
Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state

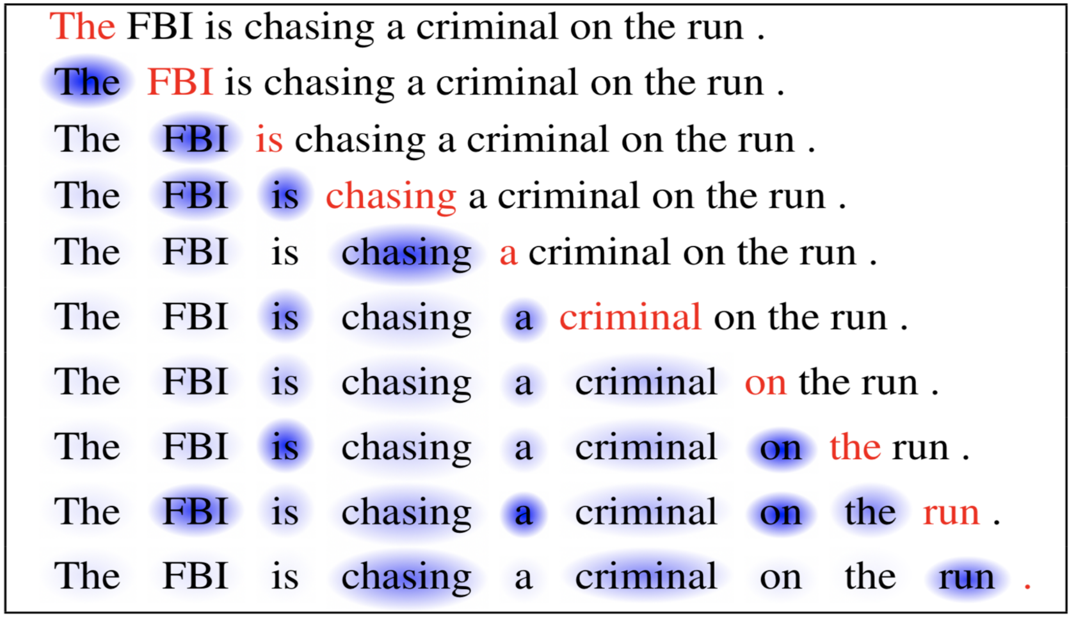


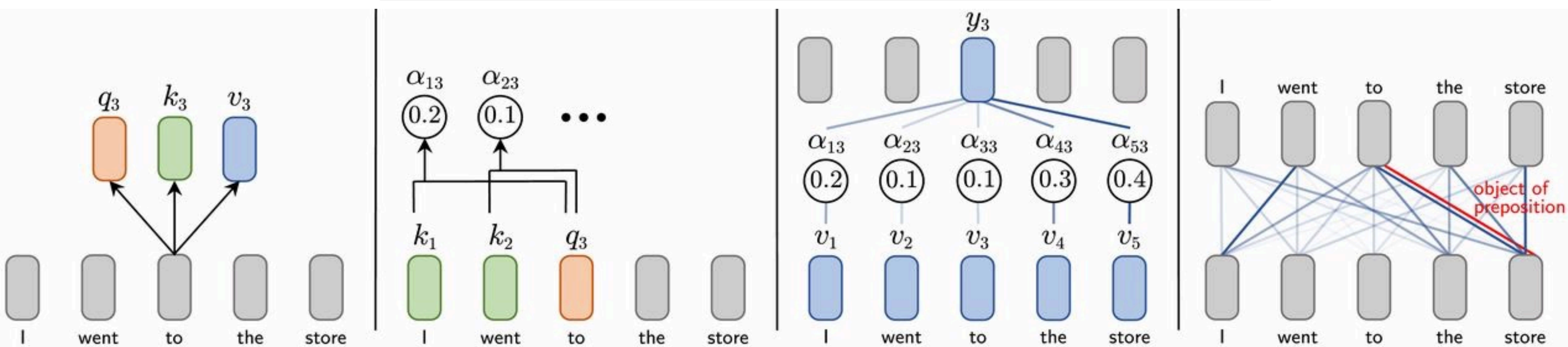
Adapt from From NIPS 2017 DL Trend Tutorial

The attention module gives us a weight for each input.



Self-attention creates attention layers mapping from a sequence to itself.





Each input vector is linearly transformed into query, key, and value vectors

Attention weights are normalized inner products of query and key vectors

Outputs are weighted sums of value vectors

After training, the attention weights can be compared with linguistic annotations

Transformer: Exploiting Self Attentions

- Uses 3 kinds of attention
  - Encoder self-attention.
  - Decoder self-attention.
  - Encoder-decoder multi-head attention.

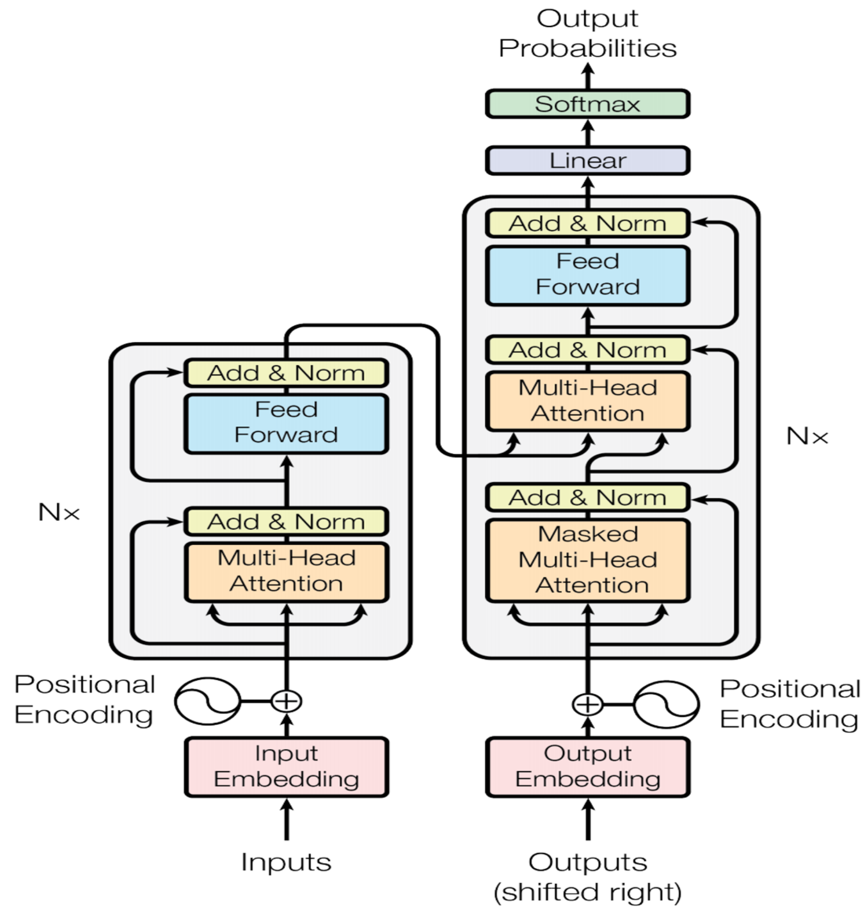
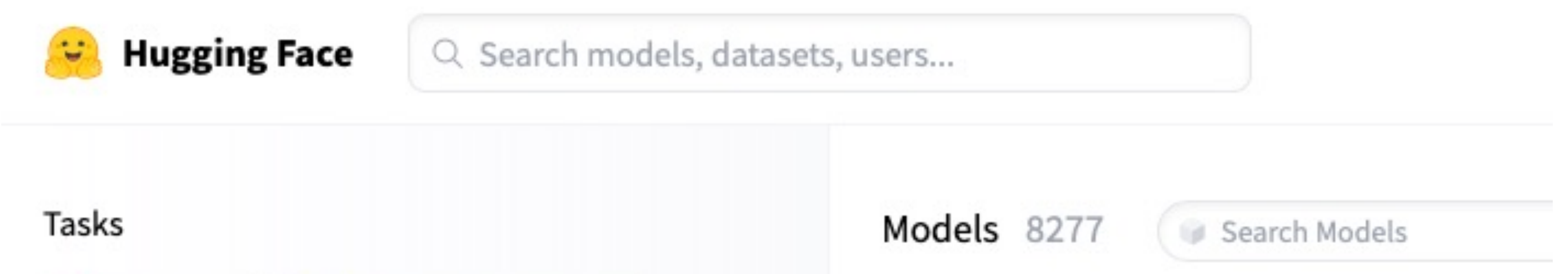
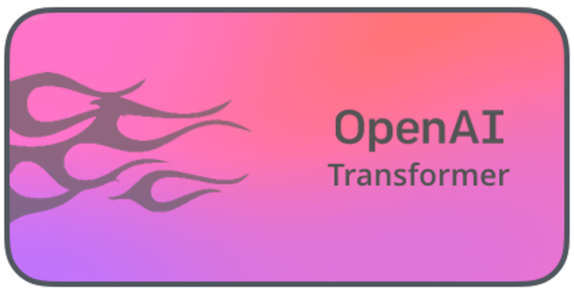


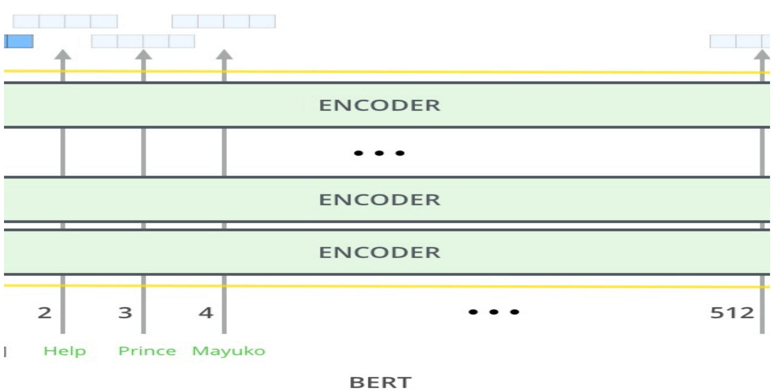
Figure 1: The Transformer - model architecture.



Notable pre-trained NLP models



BERT: Bidirectional Encoder Representations from Transformers
Pre-trained transformer encoder for sentence embedding

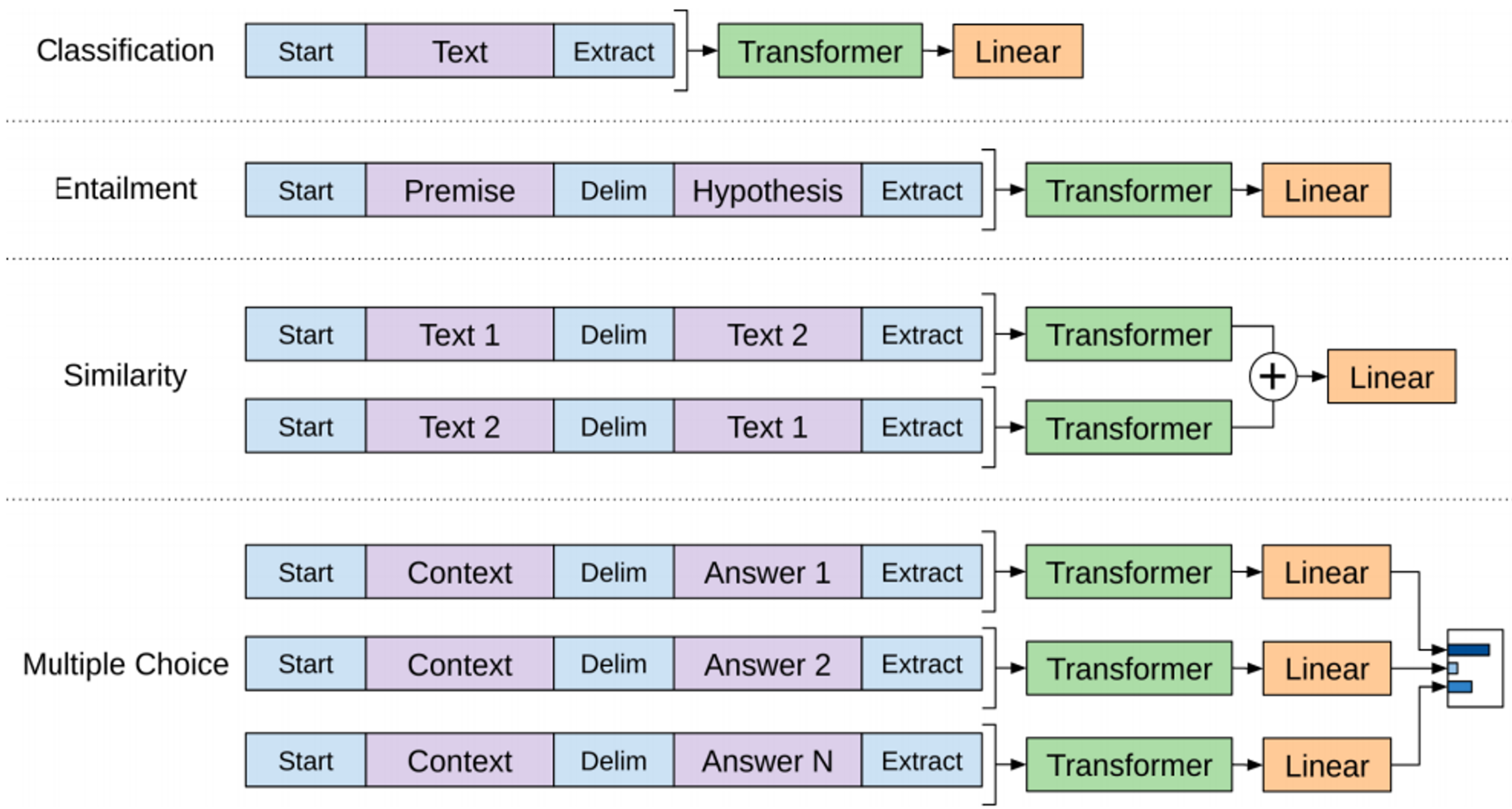


BERT's architecture is just a transformer's encoder stack.

Based: Dr. Yangqiu Song's slides

Open AI's GPT-2: 1.5 billion parameters! Trained on 8M pages from reddit

As with BERT, you can use the pretrained GPT models for any task. Different tasks use the OpenAI transformer in different ways.



GPT: generative pre-training,

GPT 's architecture is just a transformer's decoder stack.

Background:

Adversarial Examples in Vision

Background: Machine Learning

- Machine Learning: to find model F(.) that can generalize from observed data to unseen data

Input

X → Model F (.) → Output Y

X' → Model F (.) → ?

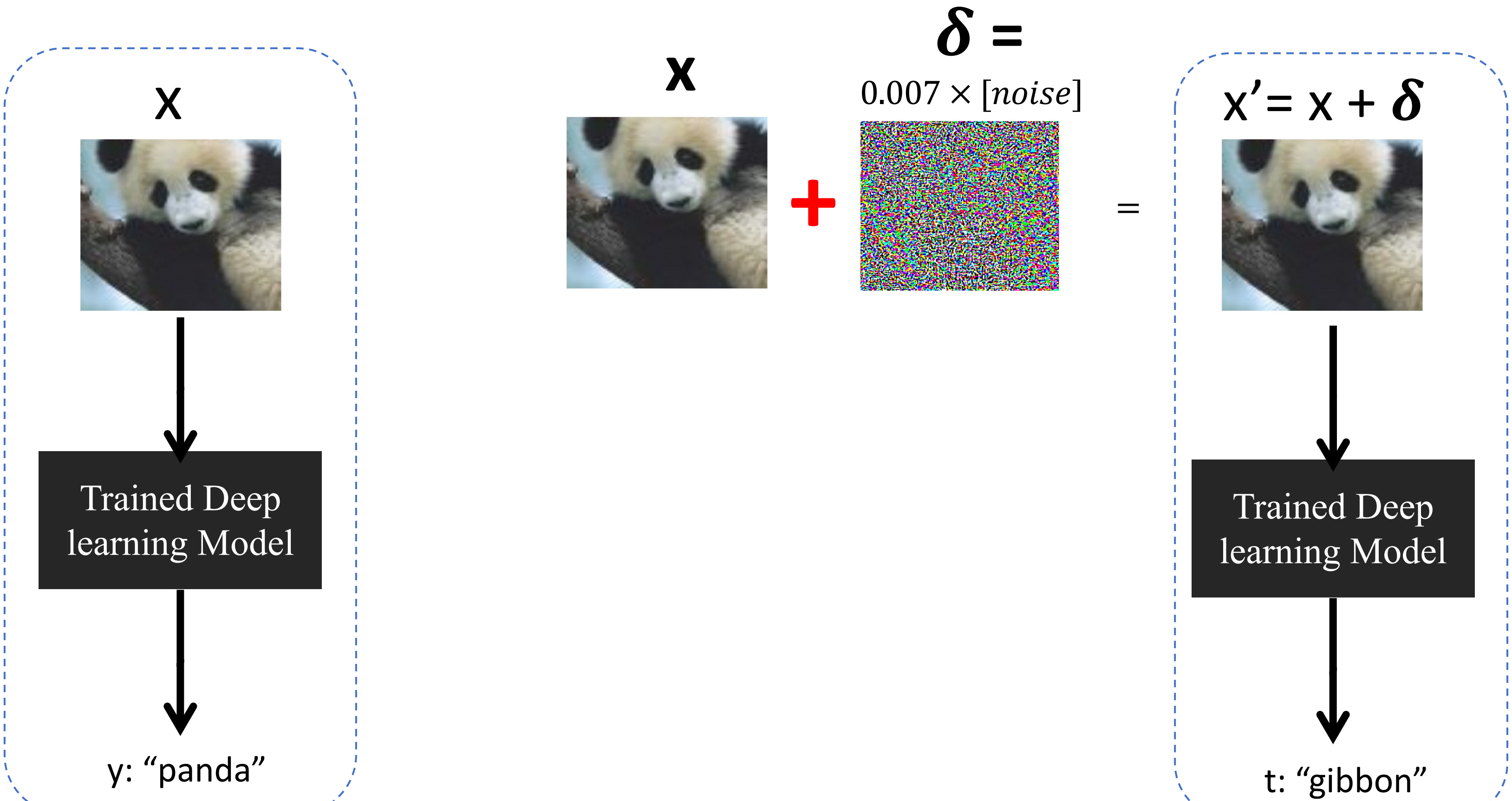
Model F (.) generalizes to Unseen X'

For instance:

Trained Deep learning Model

"panda"

Background: Adversarial Examples



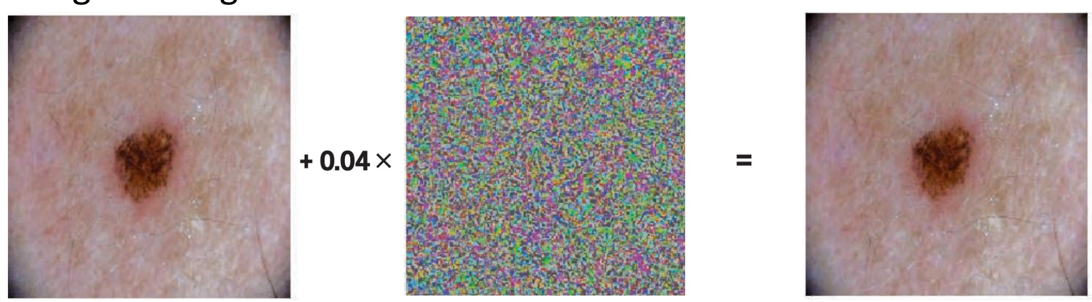
C Szegedy et al., *Intriguing Properties of Deep Neural Networks*. In *ICLR 2014*.

Deep Learning Classifiers are Easily Fooled

Healthcare

Melanoma Diagnosis with Computer Vision

Original Image | Perturbation | Adversarial Example



Benign | | **Malignant**

$\mathbf{x} + \boldsymbol{\delta}$

Samuel G Finlayson et al. “Adversarial attacks on medical machine learning”, *Science*, 2019.

Classifiers Under Attack: **Adversary Adapts**

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition

Mahmood Sharif
Carnegie Mellon University
Pittsburgh, PA, USA
mahmoods@cmu.edu

Sruti Bhagavatula
Carnegie Mellon University
Pittsburgh, PA, USA
srutib@cmu.edu

Lujo Bauer
Carnegie Mellon University
Pittsburgh, PA, USA
lbauer@cmu.edu

Michael K. Reiter
University of North Carolina
Chapel Hill, NC, USA
reiter@cs.unc.edu

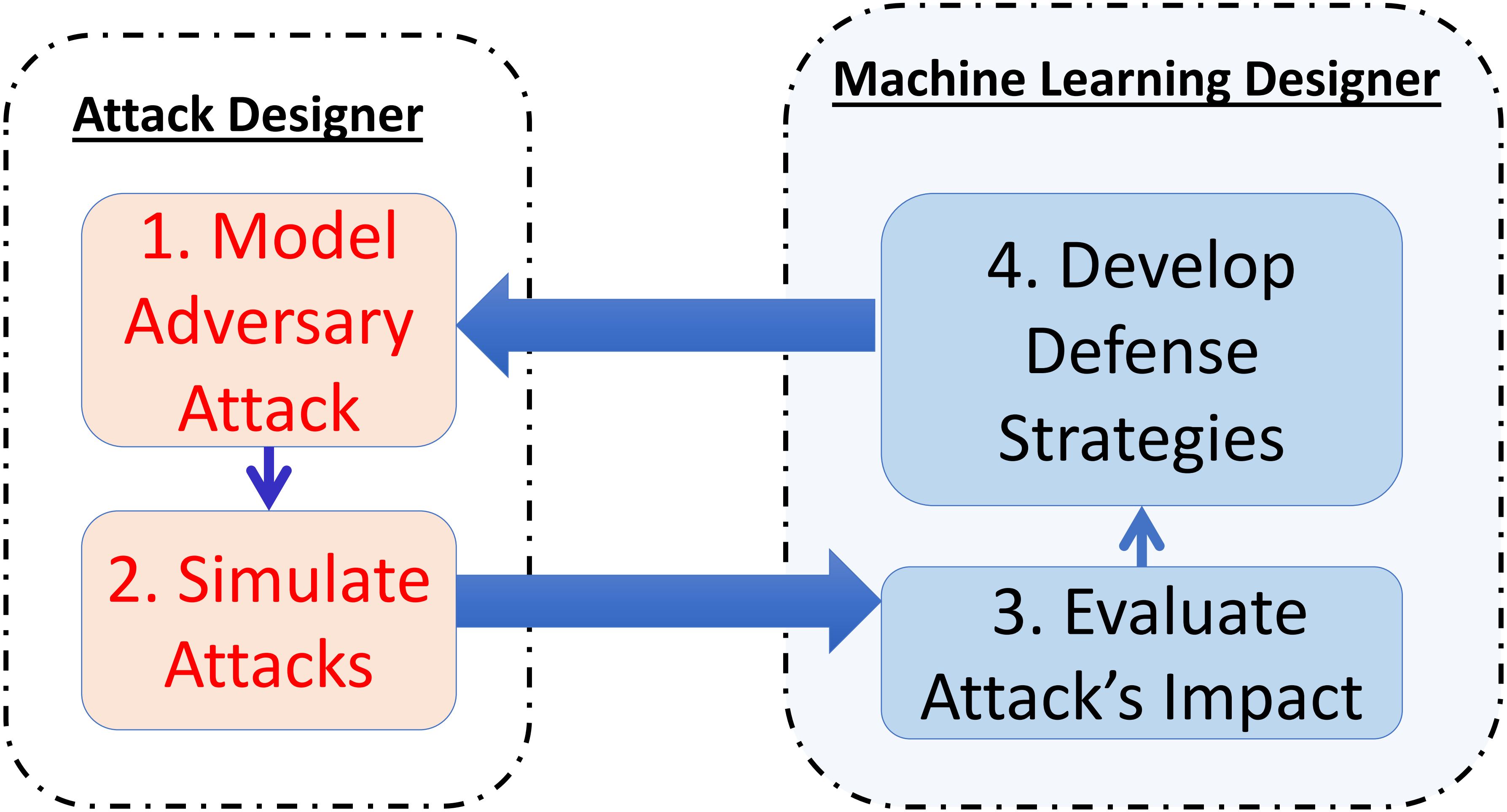
ACM CCS 2016

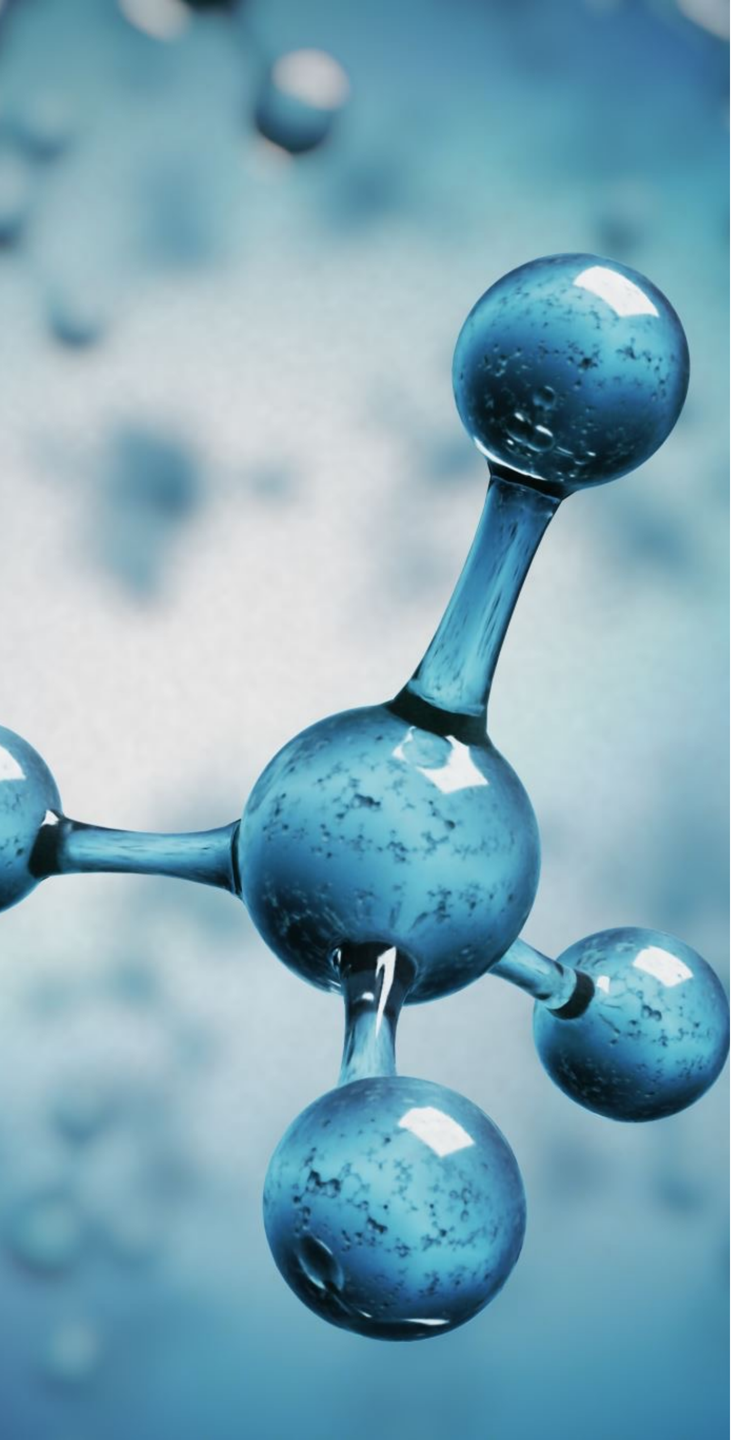
Actual images

Recognized faces

Mahmood Sharif et al. "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition", *In CCS,* 2016.

Goal of Adversarial Machine Learning





Terminology

- **Adversarial examples** or **adversarial perturbations:** changes to inputs that fool a trained model;
- We call "a program that repeatedly generates adversarial examples for a target model" as an **adversarial attack**
- We name a model's resistance to adversarial examples as **adversarial robustness**

Background: Adversarial Examples

x

Trained Deep learning Model

y: “panda”

x

+

$\boldsymbol{\delta}$ **=**

$0.007 \times [noise]$

=

x: original sample

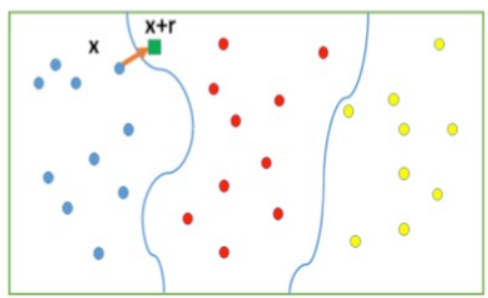


Image Space

x’ = x + $\boldsymbol{\delta}$: adversarial sample

C Szegedy et al., *Intriguing Properties of Deep Neural Networks*. In *ICLR 2014*.

x’= x + $\boldsymbol{\delta}$

Trained Deep learning Model

t: “gibbon”

How to find $\boldsymbol{\delta}$? Definition in Vision

Let F be our neural network represented as a function. To generate adversarial example for given image $x \in \mathbb{R}^m$ and its label $y \in \{1, \ldots, K\}$, we want to find some perturbation $\delta \in \mathbb{R}^m$ such that

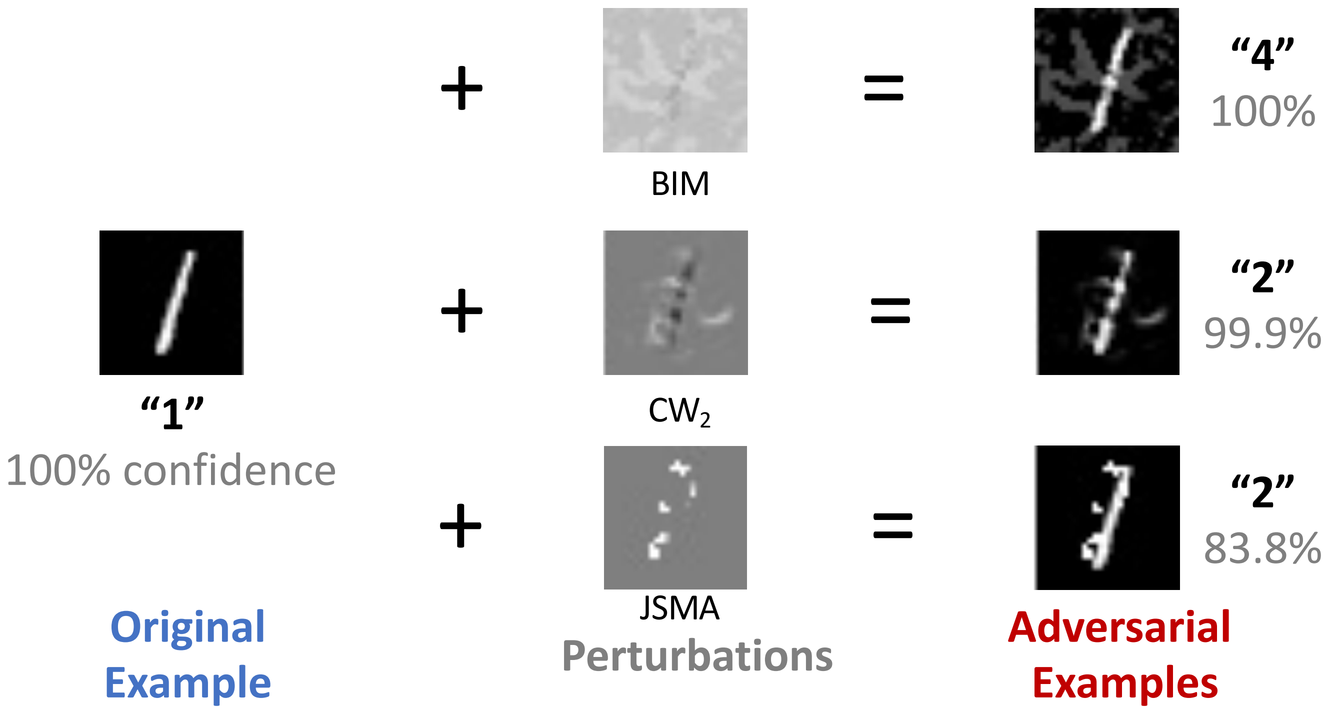
$$F(x + \delta) \neq y$$

Misclassification term

At the same time, we want $||\delta||_p \leq \epsilon$ where $\epsilon > 0$ and p is typically 2 or ∞.

Distance term

Background: Different variations of Adversarial Examples



$\mathbf{x} \quad + \quad \boldsymbol{\delta}$

C Szegedy et al., *Intriguing Properties of Deep Neural Networks*. In *ICLR 2014*.

How to search for $\boldsymbol{\delta}$?
Adversarial Attack as Optimization Problem

Generating an adversarial example therefore becomes an optimziation problem where

(3) How to define *L(.)*?

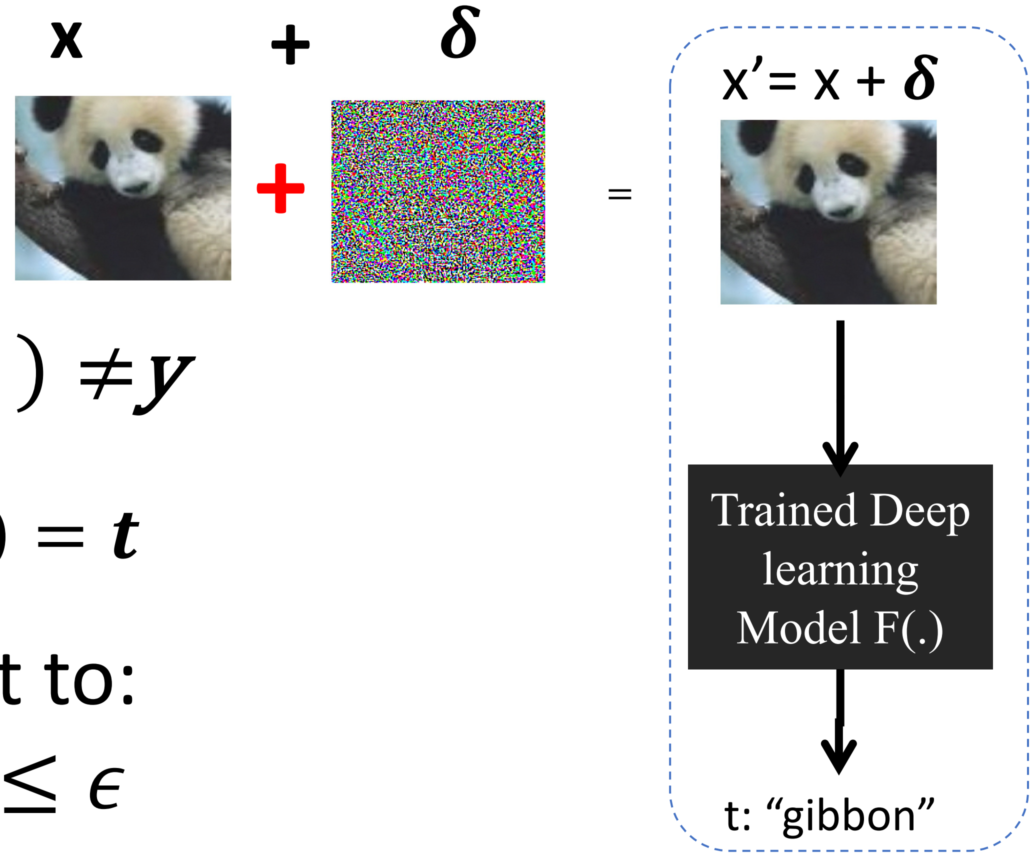
(1) How to define δ?

$$\begin{aligned} &\text{maximize} \quad L(F, x+\delta, y) \\ &\text{subject to} \quad ||\delta||_p \leq \epsilon \end{aligned}$$

(4) How to optimize this?

(2) How to limit δ?

Many different variations of formulations to search for $\boldsymbol{\delta}$?



(3) How to define *L(.)*?

untargeted $F(x + \boldsymbol{\delta}) \neq \boldsymbol{y}$

targeted $F(x + \boldsymbol{\delta}) = \boldsymbol{t}$

Subject to:

$\| \boldsymbol{\delta} \|_p \leq \epsilon$

$\epsilon > 0$

Many different variations of formulations to search for x' from x

x **+** $\boldsymbol{\delta}$

+

=

x'= x + $\boldsymbol{\delta}$

Trained Deep learning Model F(.)

t: "gibbon"

(3) How to define *L(.)*?

targeted

$$minimize \; \|F(x+\boldsymbol{\delta})-t\| + \; \lambda * \| \; \boldsymbol{\delta} \; \|_p$$

Misclassification term   Distance term

Popular Attacks in Vision: FGSM

Goodfellow et al. (2014) proposed a fast approximation algorithm called Fast Gradient Sign Method (FGSM).

Perturbation δ was found by:

(4) How to optimize to get best δ?

$$\delta = \varepsilon * \text{sign}(\nabla L(F, x, y))$$

Popular Attacks in Vision: PGD

Projected gradient descent (PGD) attack, proposed by Madry et al. (2017), can be viewed as an iterative version of FGSM.

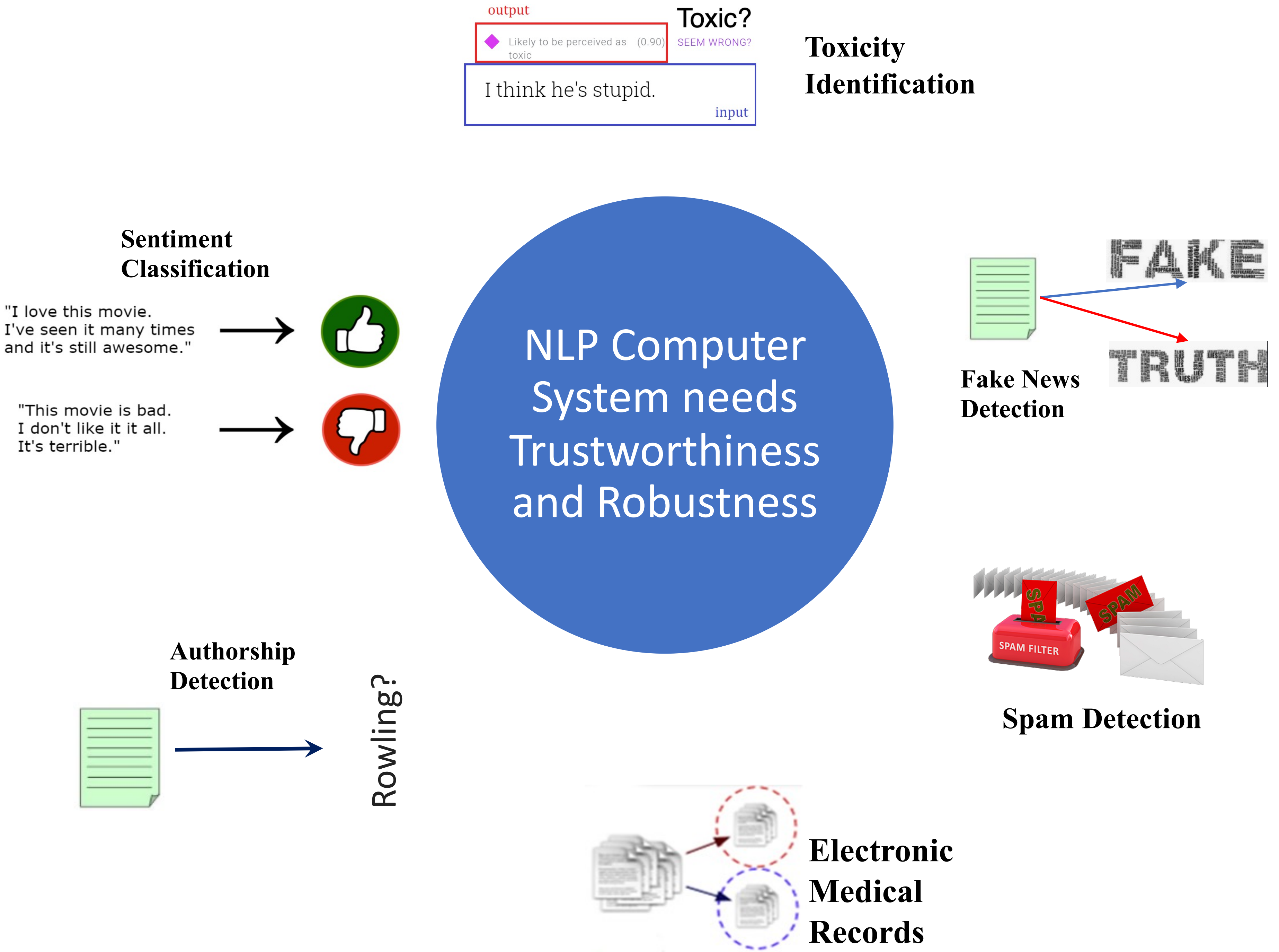
Adversarial example is found by:

(4) How to optimize to get best δ?

$$x^{t+1} = \mathsf{Clip}_{\epsilon}(x^t + \alpha * \mathsf{sign}(\nabla L(F, x^t, y))))$$

Background:

Adversarial Examples in NLP



output
Toxic?
Likely to be perceived as toxic (0.90)
SEEM WRONG?
I think he's stupid.
input
Toxicity Identification
Sentiment Classification
"I love this movie. I've seen it many times and it's still awesome."
"This movie is bad. I don't like it it all. It's terrible."
NLP Computer System needs Trustworthiness and Robustness
FAKE
TRUTH
Fake News Detection
SPAM
SPAM FILTER
Spam Detection
Authorship Detection
Rowling?
Electronic Medical Records

What about Adversarial Examples in NLP?

Naturally, we are interested if we can borrow the previous vision formulation to NLP.

But, we face some difficulties.

Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

What about AE in NLP?
Adversarial Attack as Optimization Problem

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

(3) How to define *L(.)*?

(1) How to define δ?

$$\begin{aligned} &\text{maximize} \quad L(F, x + \delta, y) \\ &\text{subject to} \quad ||\delta||_p \leq \epsilon \end{aligned}$$

(4) How to optimize this?

(2) How to limit δ?

What about AE in NLP?

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

$$\begin{array}{ll} \text{maximize} & L(F, \boxed{x + \delta}, y) \\ \text{subject to} & ||\delta||_p \leq \epsilon \end{array}$$

(1) How do you define δ for text?

What about AE in NLP?

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

$$\begin{aligned} \text{maximize} \quad & L(F, x + \delta, y) \\ \text{subject to} \quad & \boxed{||\delta||_p \leq \epsilon} \end{aligned}$$

(2) How do you limit how much to change the text?

What about AE in NLP? Adversarial Attack as Optimization Problem

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

(3) How to define *L(.)*?

$$\begin{aligned} &\text{maximize} \quad && L(F, x+\delta, y) \\ &\text{subject to} \quad && ||\delta||_p \leq \epsilon \end{aligned}$$

What about AE in NLP?
Adversarial Attack as Optimization Problem

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

$$\begin{array}{ll} \text{maximize} & L(F, x+\delta, y) \\ \text{subject to} & ||\delta||_p \leq \epsilon \end{array}$$

(4) How to optimize this?

(1) How do you define δ for NLP?

=> Four main types of perturbations

1. **Character substitution:** add, remove, or modify characters until the prediction changes.

2. **Word insertion or removal:** add or remove words until prediction changes.

3. **Paraphrase:** train a model to paraphrase sentences; iteratively paraphrase it until prediction score changes.

4. **Synonym substitution:** swap out words in the input for a direct substitution until prediction changes.

Most successful technique (so far)

$$T(x):\ \cancel{x+\boldsymbol{\delta}}$$

Transformation term

(1) How do you define δ for NLP?

- **Idea 1:** examples that are *almost* visually indistinguishable to humans (**mispellings**)

Input, $\mathbf{x}$:		**Perturbation, $\mathbf{x_{adv}}$:**
"True Grit" was the best **movie** I've seen **since** I was a small boy.	→	"True Grit" was the best **moive** I've seen **snice** I was a small boy.
Prediction: **Positive ✓**		Prediction: **Negative ✗**

Useful, but easy to defend against:

- Pass inputs into a **spell-checker** before feeding them into the model
- Or, train an RNN to correct inputs

Our Paper: Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers at 2018 IEEE Security and Privacy (SPW)

(1) How do you define δ for NLP?

- **Idea 4:** examples that are indistinguishable *in meaning* to the original input (**semantics-preserving changes**)

Input, $\mathbf{x}$:		**Perturbation, $\mathbf{x}_{\mathtt{adv}}$:**
"True Grit" was the best movie I've seen since I was a **small boy**.	→	"True Grit" was the best movie I've seen since I was a **wee lad**.
Prediction: **Positive ✓**		Prediction: **Negative ✗**

Most successful technique (so far)

AE NLP literature is messy (chaotic)

1. Many generate examples are bad
2. No standard library
3. No clear benchmarking insights (which strategy?)
4. No clear benefits

Our Solution:

TextAttack to Rescue

AE NLP literature is messy (chaotic)

1. Many generate examples are bad

(2) How do you limit how much to change from the seed text?

- Images are continuous while text is discrete. This leads to significant difference in how adversarial examples are generated.

Generating an adversarial example therefore becomes an optimziation problem where

$$\begin{aligned} \text{maximize} \quad & L(F, x+\delta, y) \\ \text{subject to} \quad & \boxed{||\delta||_p \leq \epsilon} \end{aligned}$$

(2) How do you limit how much to change the text?

Input, $\mathbf{x}$:

"True Grit" was the best movie I've seen since I was a **small boy**.

Prediction: **Positive ✓**

→

Perturbation, $\mathbf{x}_{\mathrm{adv}}$:

"True Grit" was the best movie I've seen since I was a **wee lad**.

Prediction: **Negative ✗**

Bad examples of adversarial perturbations in NLP

Perturbation, $\mathbf{x}_{\mathrm{adv}}$:

different **semantics** than original input →	"True Grit" was the **worst** movie I've seen since I was a small boy.
violates **grammar** (unlike the original input) →	"True Grit" was the best movie I've seen since I **were boy small**.
this is just **suspicious** – nobody talks like that! →	"True Grit" was the best movie I've seen since I was a **miniscule youngster**.

Constraints to ensure "valid" examples
➔ How to measure δ for NLP?

- **<u>Idea 1:</u>** what is the cosine similarity between the sentence embeddings of $\mathbf{x}$ and $\mathbf{x_{adv}}$?
  - (we can obtain sentence embeddings from the Universal Sentence Encoder, for example)
- **<u>Idea 2:</u>** Use a grammar checker to sure that we didn't introduce any grammatical errors in $\mathbf{x_{adv}}$.

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language
• 2020 [EMNLP Findings]()

all of these are TextAttack **constraints** (textattack.constraints)

Evaluation of grammar

- We evaluated syntax with **LanguageTool**, an open-source grammar checker
- Detected more errors in x_{adv} than x in 35% to 70% of samples (depending on datasets)

	(Jin et al., 2019)			(Alzantot et al., 2018)	
	IMDB	Yelp	MR	IMDB	Yelp
% with errors	61.8%	71.6%	35.5%	42.7%	43.8%

Let $T(x)$ be perturbation and $C_i(x)$ be a constraint,

$$C_1(T(x)) \land C_2(T(x)) \land \cdots \land C_m(T(x))\}$$

We propose a taxonomy of Constraints to control

Constraint	Human Evaluation Method	Automatic Evaluation Proxy
Semantic	Ask humans whether meaning is preserved	Universal Sentence Encoder
Grammatical	Ask humans to find grammatical errors in shuffled mix	LanguageTool
Character Overlap	Ask humans how similar x,x_adv look	Edit distance, BLEU, METEOR
Non-suspicious	Ask humans to identify suspicious in shuffled mix	GROVER

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language

- 2020 EMNLP Findings

Our evaluation reveals two concerns:

1. Literature's comparisons between past attacks are problematic. What is really necessary is comparison ***with the same constraints***

2. Even once constraints are standardized, researchers chose too lax thresholds! ➔We ***actually asked humans*** (via Amazon Turk) to provide guidance on the best threshold for each constraint.

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language

- 2020 EMNLP Findings

Human Study Standardized Constraints Enables Better/ Truthful Comparisons

Constraints	TFAdjusted		TextFooler	
Search Method	TextFooler	GeneticAttack	TextFooler	GeneticAttack
Semantic Preservation	4.06	4.11	-	-
Grammatical Error %	0	0	-	-
Non-suspicion Score	58.8	56.9	-	-
Attack Success %	10.6	**12.0**	91.1	**95.0**
Perturbed Word %	11.1	11.0	18.9	17.2
Num Queries	**27.1**	4431.6	**77.0**	3225.7

Table 7: Comparison of the search methods from GeneticAttack and TextFooler with two sets of constraints (TextFooler and TFAdjusted). Attacks were run on 1000 samples against BERT fine-tuned on the MR dataset. GeneticAttack's genetic algorithm is more successful than TextFooler's greedy strategy, albeit much less efficient.

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language

- 2020 [EMNLP Findings]

AE NLP literature is messy (chaotic)

2. No standard library

Problems with Current NLP Attack Ecosystem

Many attacks, but Each implemented and benchmarked in separate codebases (if released at all)

- Hard to trust literature comparisons because implementation differences can affect results
- hard to benchmark

Problems with Current NLP Attack Ecosystem

Many attacks, but Each implemented and benchmarked in [separate codebases] (if released at all)

- Hard to trust literature comparisons because implementation differences can affect results
- hard to benchmark

Challenging to develop new attacks re-using existing components

- Lots of overlap between attacks (e.g. synonym substitution techniques), but little standardization or re-usability

Problems with Current NLP Attack Ecosystem

Many attacks, but Each implemented and benchmarked in separate codebases (if released at all)	Challenging to develop new attacks re-using existing components	Difficult to utilize attacks and attack components for improving models
• Hard to trust literature comparisons because implementation differences can affect results • hard to benchmark	• Lots of overlap between attacks (e.g. synonym substitution techniques), but little standardization or re-usability	• Attack implementations are almost never model-agnostic • Adversarial training code is usually unreleased or non-existent

Generating an adversarial example therefore becomes an optimziation problem where

(3) How to define *L(.)*?

(1) How to define δ?

$$\begin{aligned} \text{maximize} \quad & L(F, x+\delta, y) \\ \text{subject to} \quad & ||\delta||_p \leq \epsilon \end{aligned}$$

(4) How to optimize this?

(2) How to limit δ?

(4) How to optimize this?

(3) Goal Function term

$T(x)$

(1) Transformation term

$$maxime: L(F, T(\boldsymbol{x}), y)$$

Subject To:

$$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$$

(2) Constraints' term

Standardize Generating NLP adversarial examples

Four Components Framework:

1. Transformation: mechanism for generating potential adversarial examples
2. Constraints: linguistic requirements for valid adversarial examples
3. Goal Function: defines end-goal for adversarial attack
4. Search Algorithm: method for finding sequence of transformations that produce valid adversarial examples defined by goal function and constraints

(4) How to optimize this?

(3) Goal Function term

$T(x)$

(1) Transformation term

$$maxime: L(F, T(\boldsymbol{x}), y)$$

Subject To:

$$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$$

(2) Constraints' term

Our Paper: TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP

•2020 EMNLP Demo

$T(x)$: Transformation term

all of these are TextAttack **transformations** (textattack.transformations)

Transformation: Word Substitution centered

- **Thesaurus:** Look up the word in a thesaurus
- **Hybrid:** Search for nearest neighbors in the ***counter-fitted*** embedding space *(Mrkšić et al, 2016)*
- **Embeddings:** Search for nearest-neighbors in the embedding space

$T(x)$: Transformation term

all of these are TextAttack **transformations** (textattack.transformations)

Transformation by Perturbing with synonyms

Lexical knowledge base

- WordNet (Miller, 1995)
- HowNet (Dong et al., 2010)

Word embedding space

- Counter-fitted
- GloVe

Masked language model

- BERT (Devlin et al., 2018)
- RoBERTa (Liu et al., 2019)

$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$

Constraints' term

all of these are TextAttack **constraints** (textattack.constraints)

Constraints

Examples:

1. **Word Embedding Similarity**: When replacing x_i with x'_i via counter-fitted word embeddings, we require that embedding of x_i and x'_i satisfy minimum cosine similarity.
2. **Part-of-speech Consistency**: To preserve fluency, we require that the two words being swapped have the same part-of-speech.
3. **Sentence Encoding Similarity**: Using sentence encoders trained for semantic textual similarity, we compare the sentence encodings of original text x and perturbed text x'.

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language

- 2020 EMNLP Findings

$L(F, T(\boldsymbol{x}), y)$ Goal Function term

TextAttack **goal functions** (textattack.goal_functions)

Goal Function:

- A way to know whether an example successfully fools the model.

Untargeted $F(x + \boldsymbol{\delta}) \neq \boldsymbol{y}$

Targeted $F(x + \boldsymbol{\delta}) = \boldsymbol{t}$

Many more special scoring for e.g. Seq2Seq outputs

TextAttack **search methods**
(textattack.search_methods)

Search Algorithm to find the best $T(\boldsymbol{x})$

- A way to **search** the space of transformations for a valid, successful adversarial example.

- Details in next section

(4) How to optimize this?

(3) Goal Function term

$T(x)$

(1) Transformation term

$$maxime: L(F, T(\boldsymbol{x}), y)$$

Subject To:

$$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$$

(2) Constraints' term

The TextAttack Framework

NLP attacks can be constructed from four components:

1. **transformation** (textattack.transformations.Transformation)
2. **constraint(s)** (list(textattack.constraints.Constraint))
3. **search method** (textattack.search_methods.SearchMethod)
4. **goal function** (textattack.goal_functions.GoalFunction)

(4) How to optimize this?

(3) Goal Function term

$T(x)$

(1) Transformation term

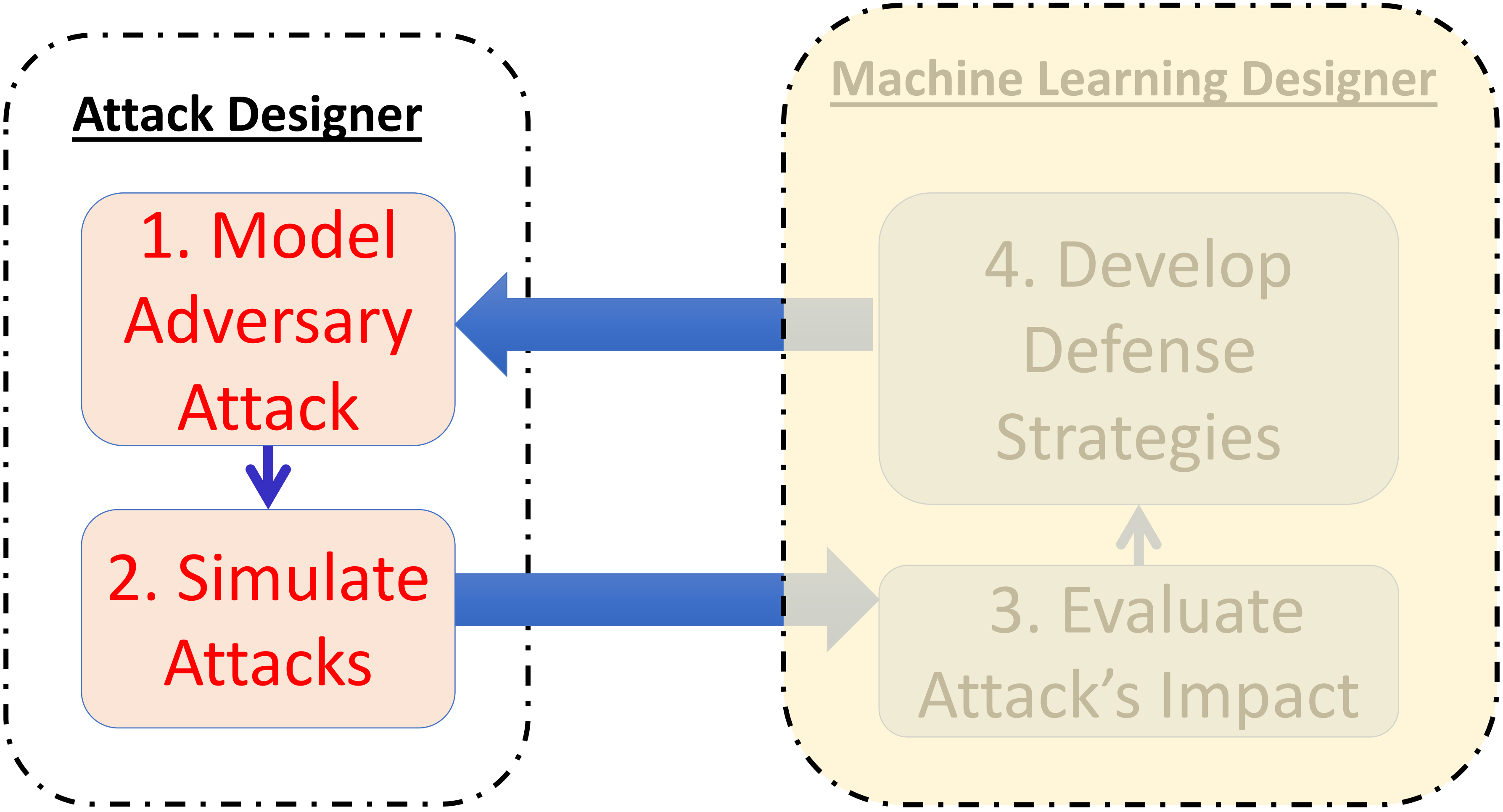
$$maxime: L(F, T(\boldsymbol{x}), y)$$

Subject To:

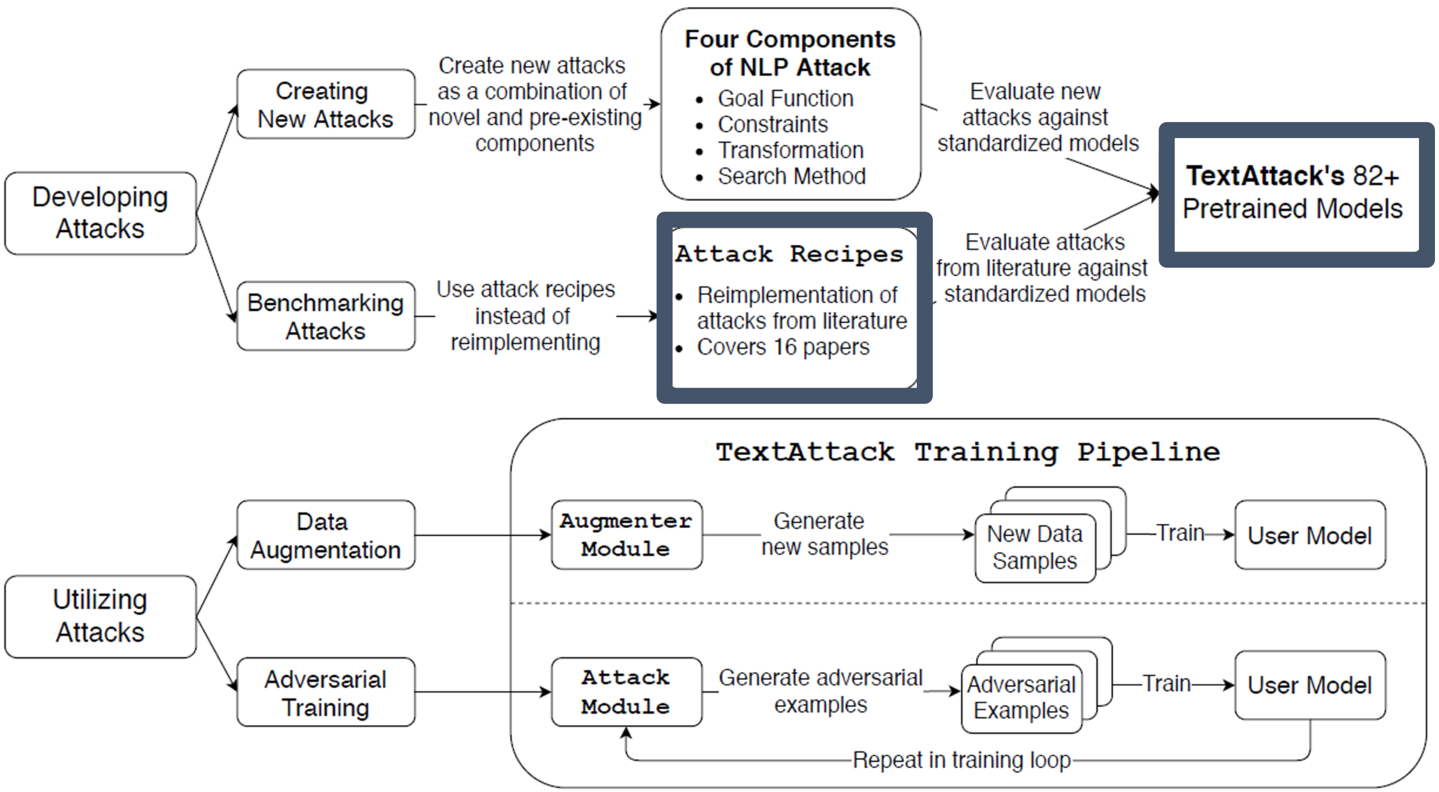
$$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$$

(2) Constraints' term

Goal of Adversarial Machine Learning



TextAttack's Features



Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment

Di Jin,[1*] Zhijing Jin,[2*] Joey Tianyi Zhou,[3] Peter Szolovits[1]
[1]Computer Science & Artificial Intelligence Laboratory, MIT
[2]University of Hong Kong
[3]A*STAR, Singapore
jindi15@mit.edu, zhijing.jin@connect.hku.hk, zhouty@ihpc.a-star.edu.sg, psz@mit.edu

Abstract

Machine learning algorithms are often vulnerable to adversarial examples that have imperceptible alterations from the original counterparts but can fool the state-of-the-art models. It is helpful to evaluate or even improve the robustness of these models by exposing the maliciously crafted adversarial examples. In this paper, we present TEXTFOOLER, a simple but strong baseline to generate adversarial text. By applying it to two fundamental natural language tasks, text classification and textual entailment, we successfully attacked three target models, including the powerful pre-trained BERT, and the widely used convolutional and recurrent neural networks. We demonstrate three advantages of this framework: (1) effective—it outperforms previous attacks by success rate and perturbation rate, (2) utility-preserving—it preserves semantic content, grammaticality, and correct types classified by humans, and (3) efficient—it generates adversarial text with computational complexity linear to the text length.[1]

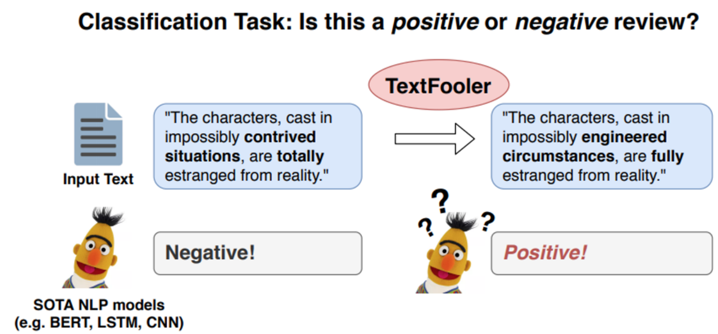


Figure 1: Our model TextFooler slightly change the input text but completely altered the prediction result.

al. 2013; Carlini and Wagner 2018), it is still challenging to deal with text data due to its discrete nature. Formally, besides the ability to fool the target models, outputs of a natural

Algorithm 1 Adversarial Attack by TEXTFOOLER

Input: Sentence example $X = \{w_1, w_2, ..., w_n\}$, the corresponding ground truth label Y, target model F, sentence similarity function $\mathrm{Sim}(\cdot)$, sentence similarity threshold ϵ, word embeddings Emb over the vocabulary Vocab.

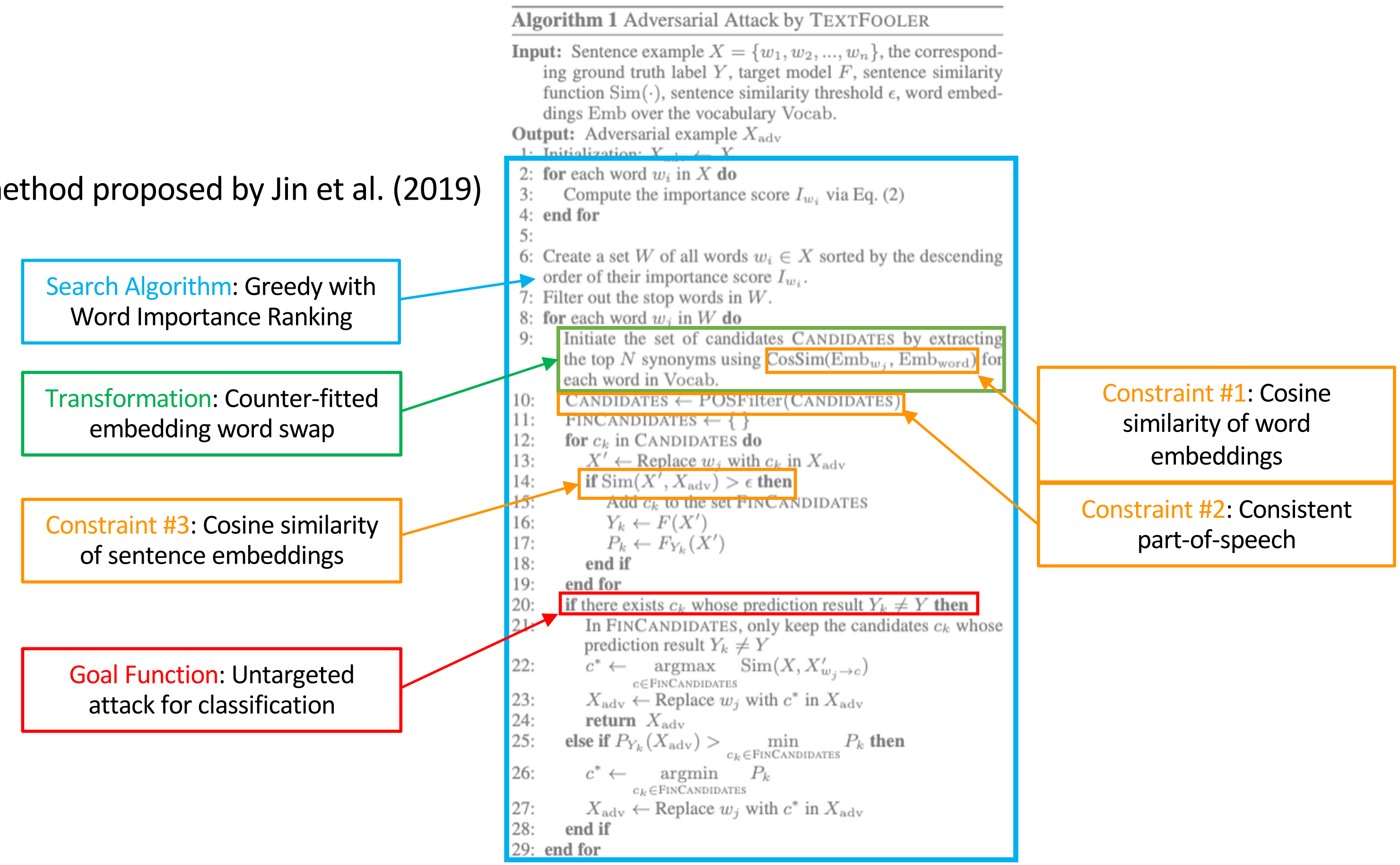
Output: Adversarial example X_{adv}

Initialization: $X_{\mathrm{adv}} \leftarrow X$
for each word w_i in X **do**
    Compute the importance score I_{w_i} via Eq. (2)
end for

Create a set W of all words $w_i \in X$ sorted by the descending order of their importance score I_{w_i}.
Filter out the stop words in W.
for each word w_j in W **do**
    Initiate the set of candidates CANDIDATES by extracting the top N synonyms using CosSim(Emb_{w_j}, $\mathrm{Emb}_{\mathrm{word}}$) for each word in Vocab.
    $\mathrm{CANDIDATES} \leftarrow \mathrm{POSFilter}(\mathrm{CANDIDATES})$
    $\mathrm{FINCANDIDATES} \leftarrow \{\ \}$
    for c_k in CANDIDATES **do**
        $X' \leftarrow$ Replace w_j with c_k in X_{adv}
        if $\mathrm{Sim}(X', X_{\mathrm{adv}}) > \epsilon$ **then**
            Add c_k to the set FINCANDIDATES
            $Y_k \leftarrow F(X')$
            $P_k \leftarrow F_{Y_k}(X')$
        end if
    end for
    if there exists c_k whose prediction result $Y_k \neq Y$ **then**
        In FINCANDIDATES, only keep the candidates c_k whose prediction result $Y_k \neq Y$
        $c^* \leftarrow \underset{c \in \mathrm{FINCANDIDATES}}{\mathrm{argmax}} \; \mathrm{Sim}(X, X'_{w_j \rightarrow c})$
        $X_{\mathrm{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
        return X_{adv}
    else if $P_{Y_k}(X_{\mathrm{adv}}) > \underset{c_k \in \mathrm{FINCANDIDATES}}{\min} P_k$ **then**
        $c^* \leftarrow \underset{c_k \in \mathrm{FINCANDIDATES}}{\mathrm{argmin}} \; P_k$
        $X_{\mathrm{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
    end if
end for
return `None`

Four Components in Action

TextFooler method proposed by Jin et al. (2019)



Algorithm 1 Adversarial Attack by TextFooler

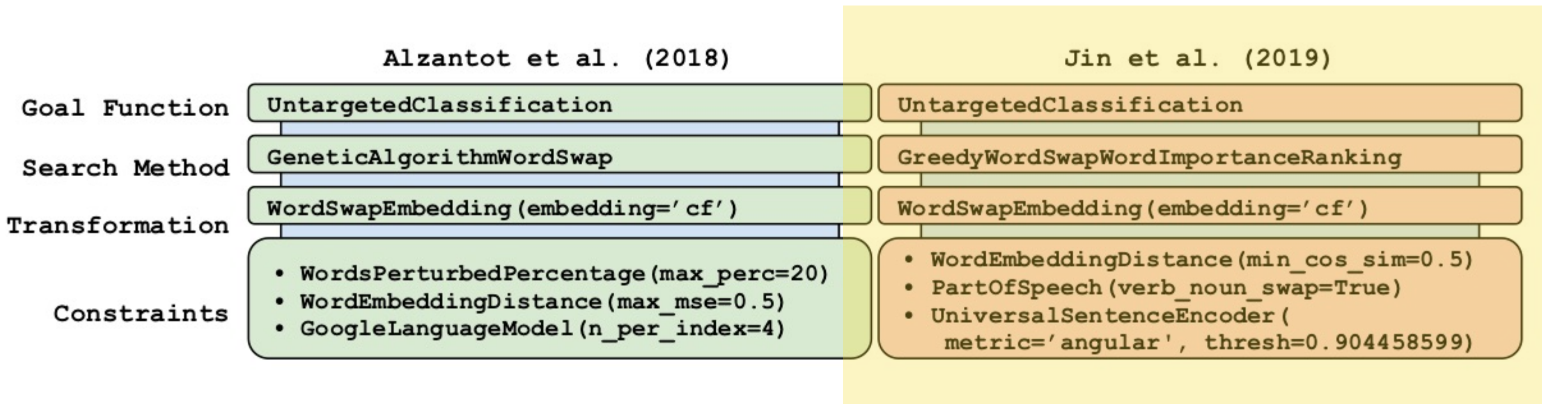
Input: Sentence example $X = \{w_1, w_2, ..., w_n\}$, the corresponding ground truth label Y, target model F, sentence similarity function $\mathrm{Sim}(\cdot)$, sentence similarity threshold ϵ, word embeddings Emb over the vocabulary Vocab.

Output: Adversarial example X_{adv}

```
Initialization: X_adv ← X
for each word w_i in X do
    Compute the importance score I_{w_i} via Eq. (2)
end for

Create a set W of all words w_i ∈ X sorted by the descending order of their importance score I_{w_i}.
Filter out the stop words in W.
for each word w_j in W do
    Initiate the set of candidates CANDIDATES by extracting the top N synonyms using CosSim(Emb_{w_j}, Emb_word) for each word in Vocab.
    CANDIDATES ← POSFilter(CANDIDATES)
    FINCANDIDATES ← { }
    for c_k in CANDIDATES do
        X' ← Replace w_j with c_k in X_adv
        if Sim(X', X_adv) > ε then
            Add c_k to the set FINCANDIDATES
            Y_k ← F(X')
            P_k ← F_{Y_k}(X')
        end if
    end for
    if there exists c_k whose prediction result Y_k ≠ Y then
        In FINCANDIDATES, only keep the candidates c_k whose prediction result Y_k ≠ Y
        c* ← argmax_{c ∈ FINCANDIDATES} Sim(X, X'_{w_j → c})
        X_adv ← Replace w_j with c* in X_adv
        return X_adv
    else if P_{Y_k}(X_adv) > min_{c_k ∈ FINCANDIDATES} P_k then
        c* ← argmin_{c_k ∈ FINCANDIDATES} P_k
        X_adv ← Replace w_j with c* in X_adv
    end if
end for
```

Four Components Standardized 18 Attacks:

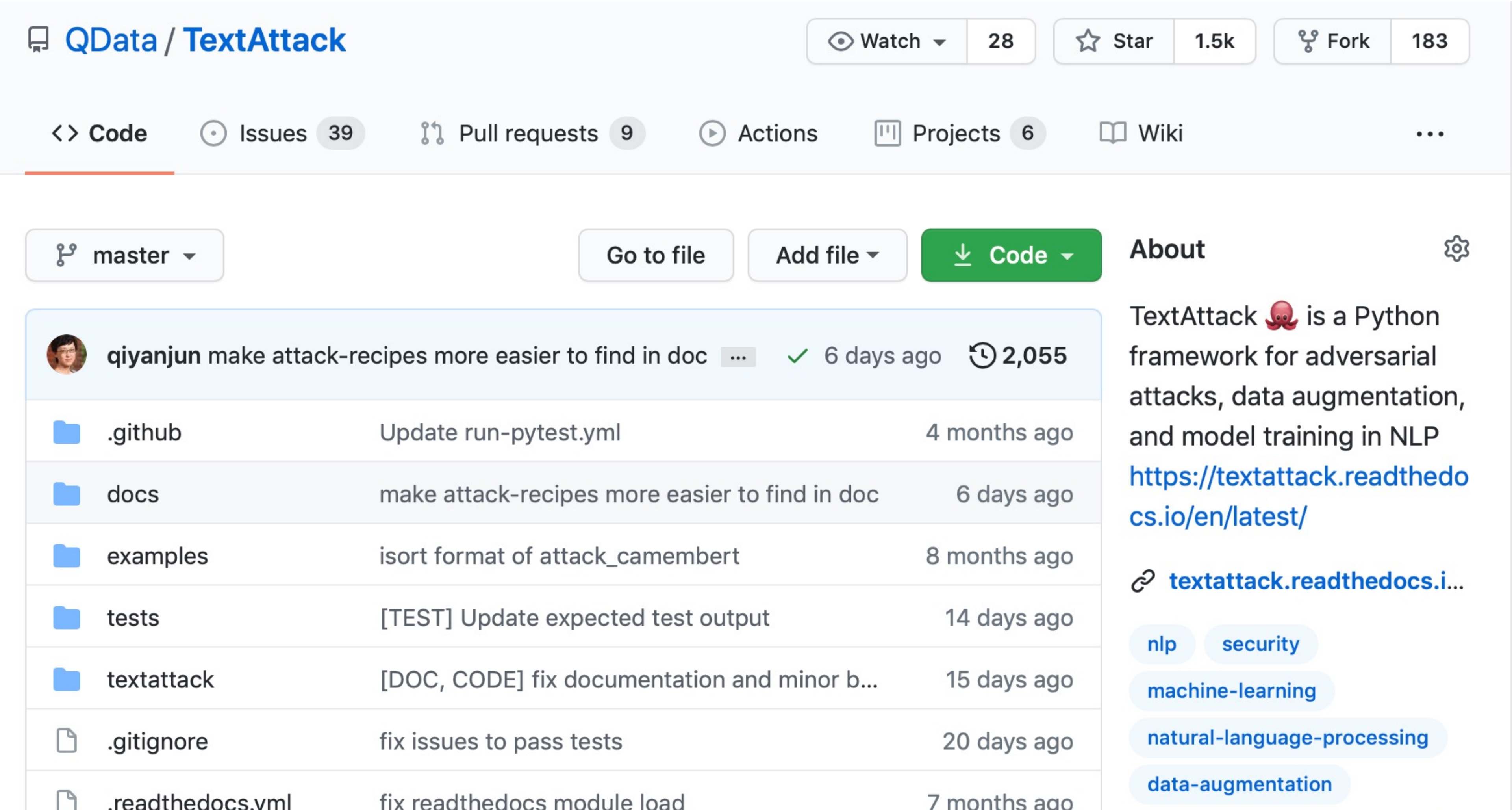


Installing TextAttack

Github PyTest passing | pypi package 0.2.12

pip install textattack

https://github.com/QData/TextAttack



We have also shared 82 Pretrained Models

Integration with HuggingFace's Model Hub and nlp library

- Can attack any model on the model hub on any dataset from nlp

TextAttack has 82 pretrained models on its Model Hub page

- Models: BERT, DistilBERT, ALBERT, BART, RoBERTa, XLNet
- Trained on all GLUE tasks

TextAttack 🐙

An open-source, model-agnostic library for **attacking NLP models** and **standardizing evaluations**

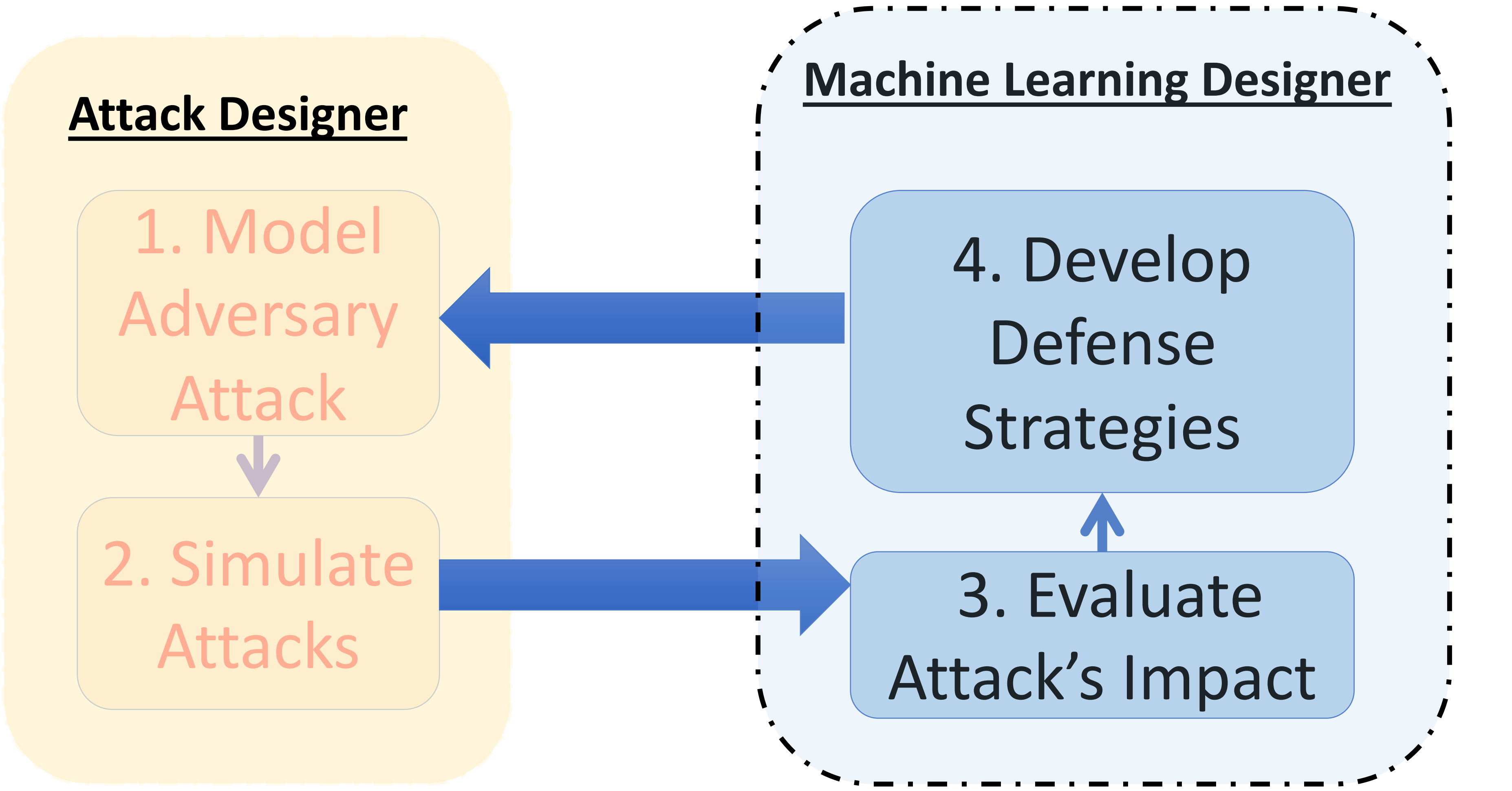
Contains 18 popular white-box and black-box attacks

82 Pre-trained models on popular datasets across multiple Types of NLP tasks

AE NLP literature is messy (chaotic)

3. No clear benchmarking insights

Goal of Adversarial Machine Learning



Search Method

Typically, one word replacement is not enough to change the model's prediction. Instead, a set of word replacements is necessary.

(4) How to optimize this?

(3) Goal Function term

$T(x)$

(1) Transformation term

$$maxime: L(F, T(\boldsymbol{x}), y)$$

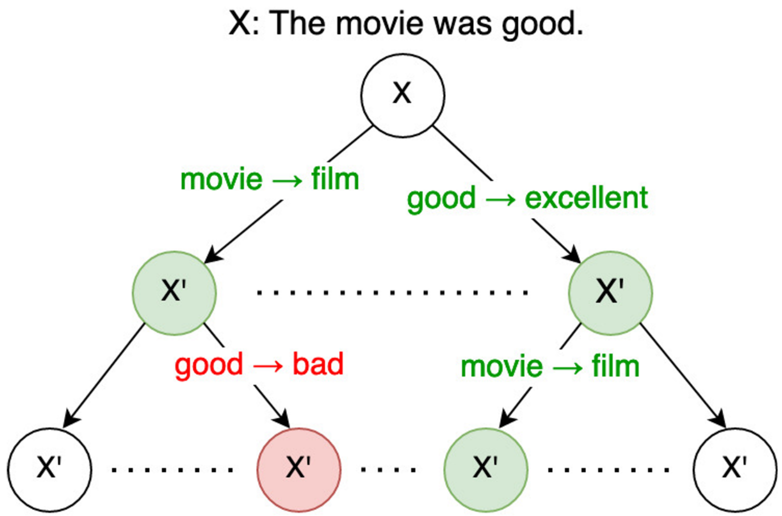
Subject To:

$$C_1(T(x)) \wedge C_2(T(x)) \wedge \cdots \wedge C_m(T(x))\}$$

(2) Constraints' term

Search Method

Typically, one word replacement is not enough to change the model's prediction. Instead, a set of word replacements is necessary.

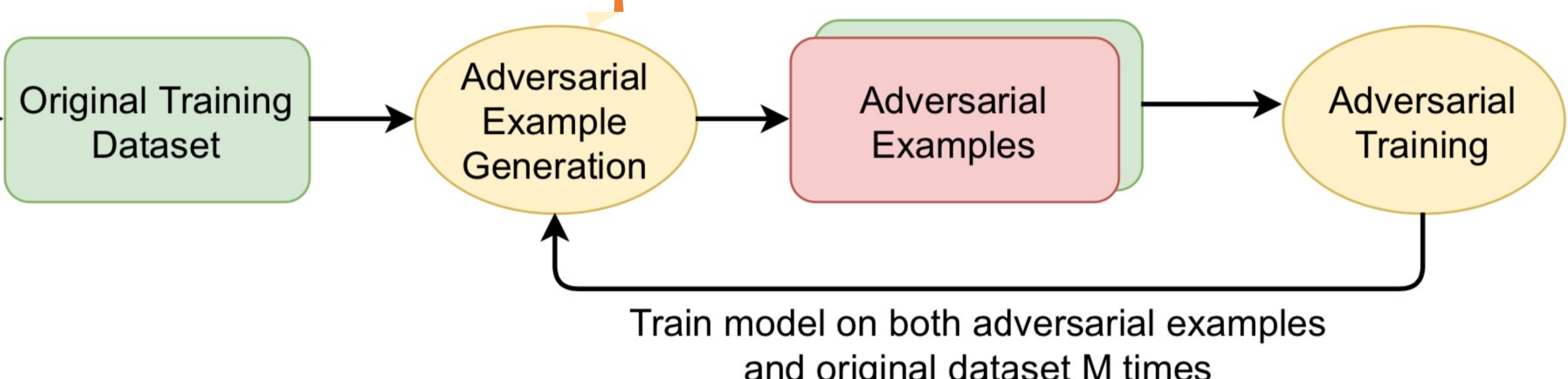


Motivation

- Adversarial training uses both clean examples and adversarial examples to train robust models.
- Two criteria we need to consider when constructing attacks for adversarial training are:

  1.Speed

  2.Capability
- Search method controls the natural trade-off between speed and capacity.

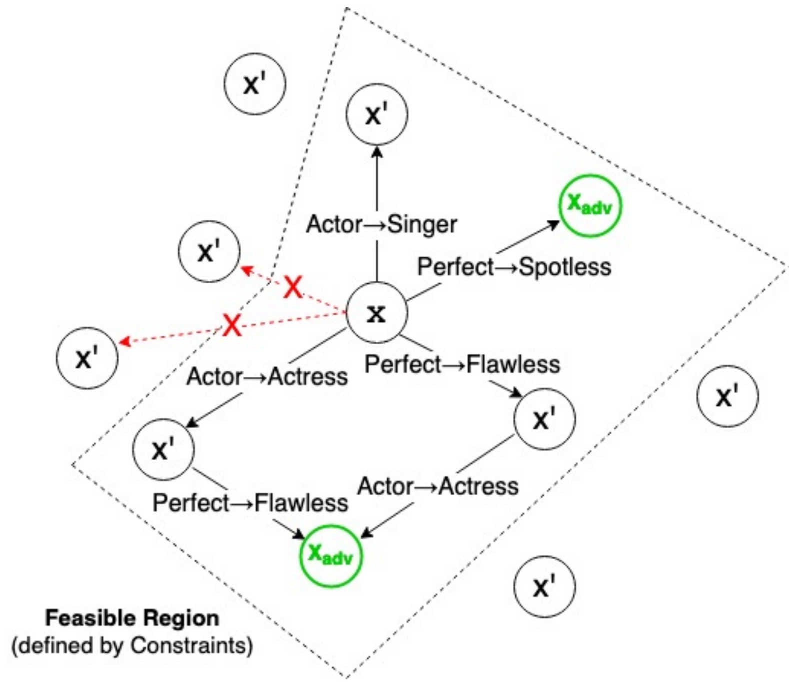


Search Algorithm: A way to **search** the space of transformations for a valid, successful adversarial example.

Search Algorithm

Why a search algorithm?

- We need to find set of transformations that successfully produce x_{adv}
- Combinatorial search problem with heuristic $score(x)$ provided by goal function



Our Analysis paper: Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples

•2020 EMNLP BlackBoxNLP

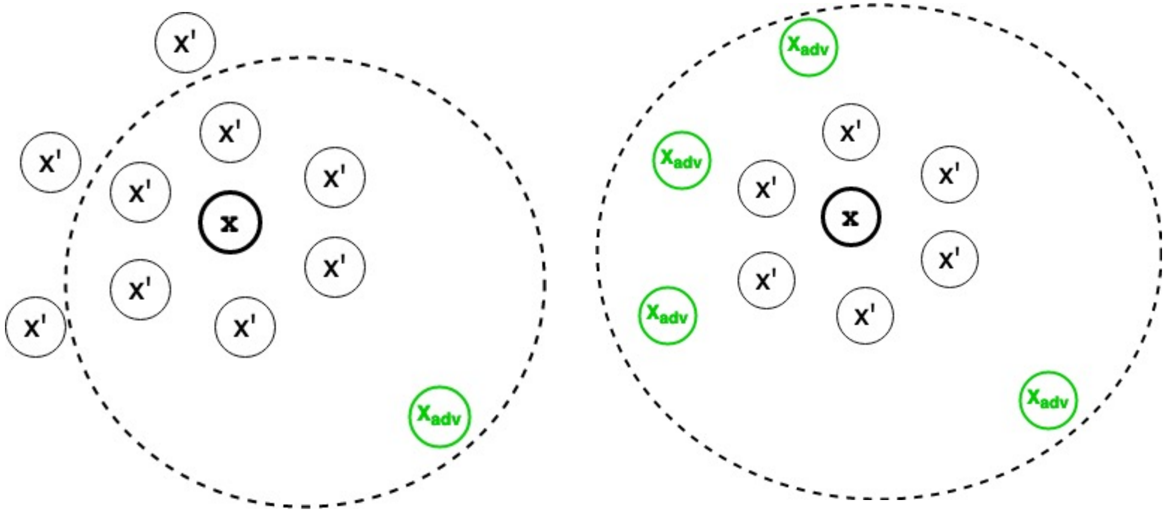
Search Algorithm: A way to **search** the space of transformations for a valid, successful adversarial example.

Search Space

Search space defined by transformation and constraints

Let $T(x)$ be our transformation and $C_i(x)$ be a constraint,

$$S(x) = \{T(x) | C_1(T(x)) \land C_2(T(x)) \land \cdots \land C_m(T(x))\}$$



Strict constraints

Lax constraints

Our Analysis paper: Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples

•2020 EMNLP BlackBoxNLP

Search Algorithms from Literature

Beam Search (Ebrahimi et al., 2017)

Greedy with Word Importance Ranking

- UNK (Gao et al., 2018)
- DEL (Jin et al., 2019)
- PWWS (Ren et al., 2019)

Genetic Algorithm (Alzantot et al., 2018),

Particle Swarm Optimization (Zang et al., 2020)

Problems in Current Literature

Inconsistent search space for comparisons

Lack of comprehensive *performance* benchmark for search algorithm

Lack of comprehensive *speed* benchmark for search algorithm

Our Analysis paper: Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples

•2020 EMNLP BlackBoxNLP

Benchmarking Insights

Optimal search for absolute performance is beam search with beam width of 8.

When within a small query budget, greedy with word importance ranking is most effective

For two constraint settings across three datasets, the relative differences between the attack success rates of greedy with word importance ranking and the success rates of beam search are less than 20%.

If only aiming for attack success, Search algorithms matter less than transformations and constraints.

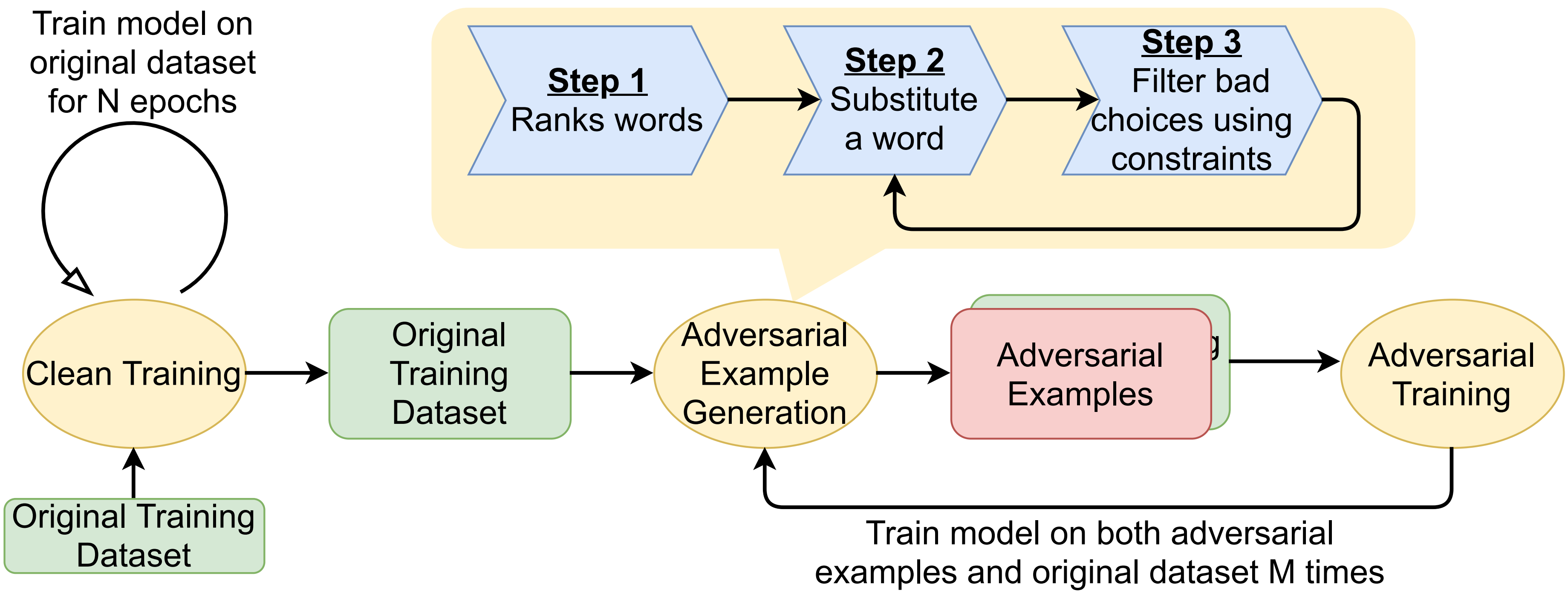
Although changing the search methods did not change attack success rate by more than 20%, changing the constraints changed attack success rate by over 60%.

Our Analysis paper: Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples

•2020 EMNLP BlackBoxNLP

AE NLP literature is messy (chaotic)

4. No clear benefits


Train model on original dataset for N epochs
Step 1
Ranks words
Step 2
Substitute a word
Step 3
Filter bad choices using constraints
Clean Training
Original Training Dataset
Adversarial Example Generation
Adversarial Examples
Adversarial Training
Original Training Dataset
Train model on both adversarial examples and original dataset M times

Adversarial Training in recent NLP Literature

Collection of recent works on **adversarial training** do not study whether it defends against **adversarial attacks** proposed in literature!

Adversarial Training in recent NLP Literature

Collection of recent works on **adversarial training** do not study whether it defends against **adversarial attacks** proposed in literature!

Adversarial Training

- *FreeLB* (Zhu et al., 2019)
- *SMART* (Jiang et al., 2019)
- *ALUM* (Liu et al., 2020)

Adversarial Training in recent NLP Literature

Collection of recent works on **adversarial training** do not study whether it defends against **adversarial attacks** proposed in literature!

Adversarial Training

- *FreeLB* (Zhu et al., 2019)
- *SMART* (Jiang et al., 2019)
- *ALUM* (Liu et al., 2020)

Adversarial Attacks

- Alzantot et al. (2018)
- *TextFooler* (Jin et al., 2019)
- PWWS (Ren et al., 2019)
- Zang et al. (2020)
- *BAE* (Garg and Ramakrishnan, 2020)
- *BERT-Attack* (Li et al., 2020)
- *CLARE* (Li et al., 2021)

Adversarial Training in NLP

Collection of recent works on **adversarial training** do not study whether it defends against **adversarial attacks** proposed in literature!

Adversarial Training

- *FreeLB* (Zhu et al., 2019)
- *SMART* (Jiang et al., 2019)
- *ALUM* (Liu et al., 2020)

?

Adversarial Attacks

- Alzantot et al. (2018)
- *TextFooler* (Jin et al., 2019)
- PWWS (Ren et al., 2019)
- Zang et al. (2020)
- *BAE* (Garg and Ramakrishnan, 2020)
- *BERT-Attack* (Li et al., 2020)
- *CLARE* (Li et al., 2021)

Recent NLP Adversarial Training (above) add perturbations in the embedding, instead of in the input space; They don't evaluate robustness well.

Adversarial Training for Robustness

Adversarial loss

Regular loss

Goodfellow et al. (2015):

$$\arg\min_{\theta}\Big[\mathbb{E}_{(x,y)\in\hat{p}_{\text{data}}}\Big(\max_{\delta\in S} L(\theta, x+\delta, y)\Big) + \mathbb{E}_{(x,y)\in\hat{p}_{\text{data}}}\Big(L(\theta, x, y)\Big)\Big] \quad (2)$$

Madry et al. (2017):

$$\arg\min_{\theta}\mathbb{E}_{(x,y)\in\hat{p}_{\text{data}}}\Big(\max_{\delta\in S} L(\theta, x+\delta, y)\Big) \quad (1)$$

Kannan et al. (2018):

$$J(\mathbb{M}, \boldsymbol{\theta}) + \lambda\frac{1}{m}\sum_{i=1}^{m} L\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), f(\tilde{\boldsymbol{x}}^{(i)}; \boldsymbol{\theta})\right).$$

Adversarial loss

Adversarial Logit Pairing

We propose faster attacks that suit for vanilla adversarial training.

- Many engineering tricks to make vanilla adversarial training feasible in NLP

- We observe that
  - Adversarial training can help improve adversarial robustness against attacks that were *not used to trained the model*.
  - Adversarial training can provide a regularization effect and improve the model's *standard accuracy* and *cross-domain generalization*.
  - Adversarial training can improve the model's ***interpretability.***

Our Analysis paper on Adversarial Training for Robust NLP Models

•2021 Under review

TextAttack Rescues Messy AE NLP literature

1. Many generate examples are bad

2. No standard library

3. No clear benchmarking insights

4. No clear benefits

Who is TextAttack for?

- **researchers** who want to implement new NLP attacks or compare them in a standardized framework

- **any machine learning practitioner** who want to understand their limitations of NLP models and/or use adversarial training to make their models better

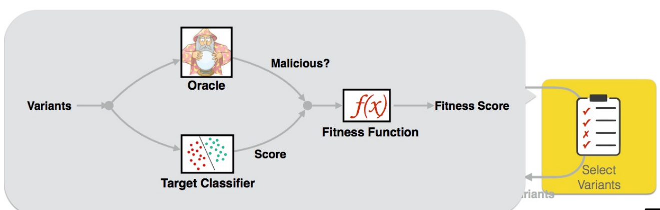
- **anyone training an NLP model** who wants to apply data augmentation to increase test-set accuracy by 1-2%

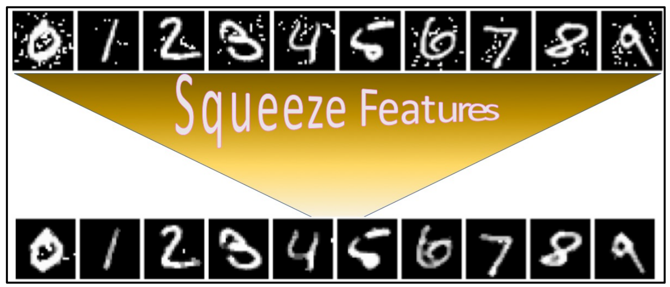
http://trustworthymachinelearning.org

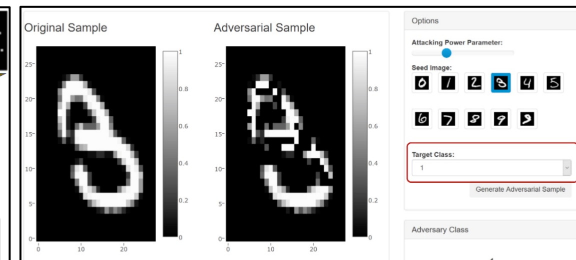
1. To Fool / Evade Learned Models	**2. To Detect fooling/ Evasion**	**3. To Defend Against Evasion**	**4. To Visualize and Benchmarking**	**5. To Understand Theoretically**

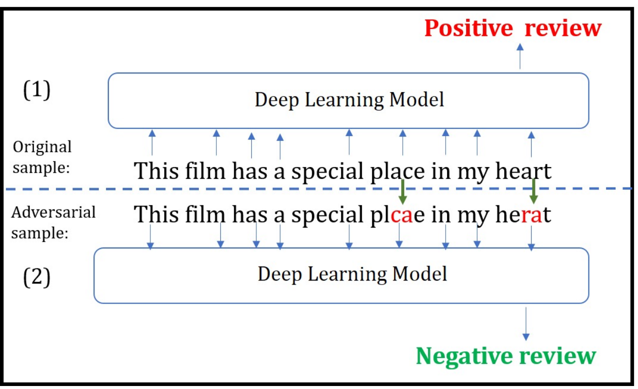
Automated Evasion Approach

Based on Genetic Programming

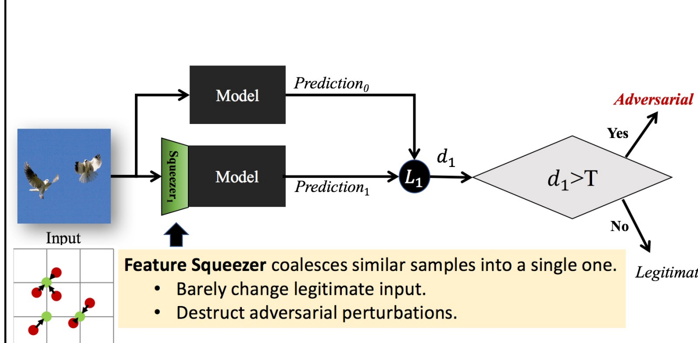


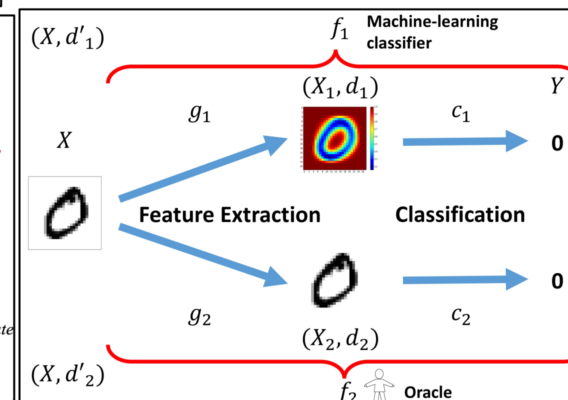






Detection Framework

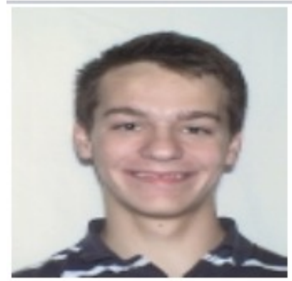
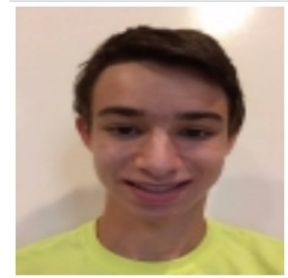




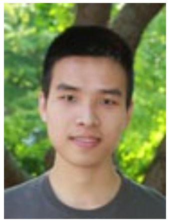
http://trustworthymachinelearning.org/

Timeline of Our Robust DL front

Acknowledgements


My Students on this project:



UVA Computer Science Dept. Security Research Group: Prof. David Evans

UVA Computer Science Dept. NLP Research Group: Prof. Yangfeng Ji

UVA Computer Science Dept. Software Safety Group: Prof. Matthew B Dwyer

UVA Computer Science Dept. Software Engineering Group: Prof. Mary Lou Soffa

Thank you

What can I do with TextAttack?

- **run standardized attack recipes** on models & datasets (yours our ours)
- visualize attack results using the command line, Visdom, W&B, etc.
- or, use the infrastructure of TextAttack to develop and benchmark **your own NLP attacks**
- or, use the components of TextAttack for **adversarial training**
- or, use the components from TextAttack for **data augmentation**