TextAttack:

Generalizing Adversarial Examples to Natural Language Processing

@ UVA Human and Machine Intelligence Seminar
2021/04/13

Yanjun Qi

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

TextAttack:

Generalizing Adversarial Examples to Natural Language Processing

@ UVA Human and Machine Intelligence Seminar
2021/04/13

Yanjun Qi

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

Background:

Natural Language Processing and Recent advances by Deep Learning

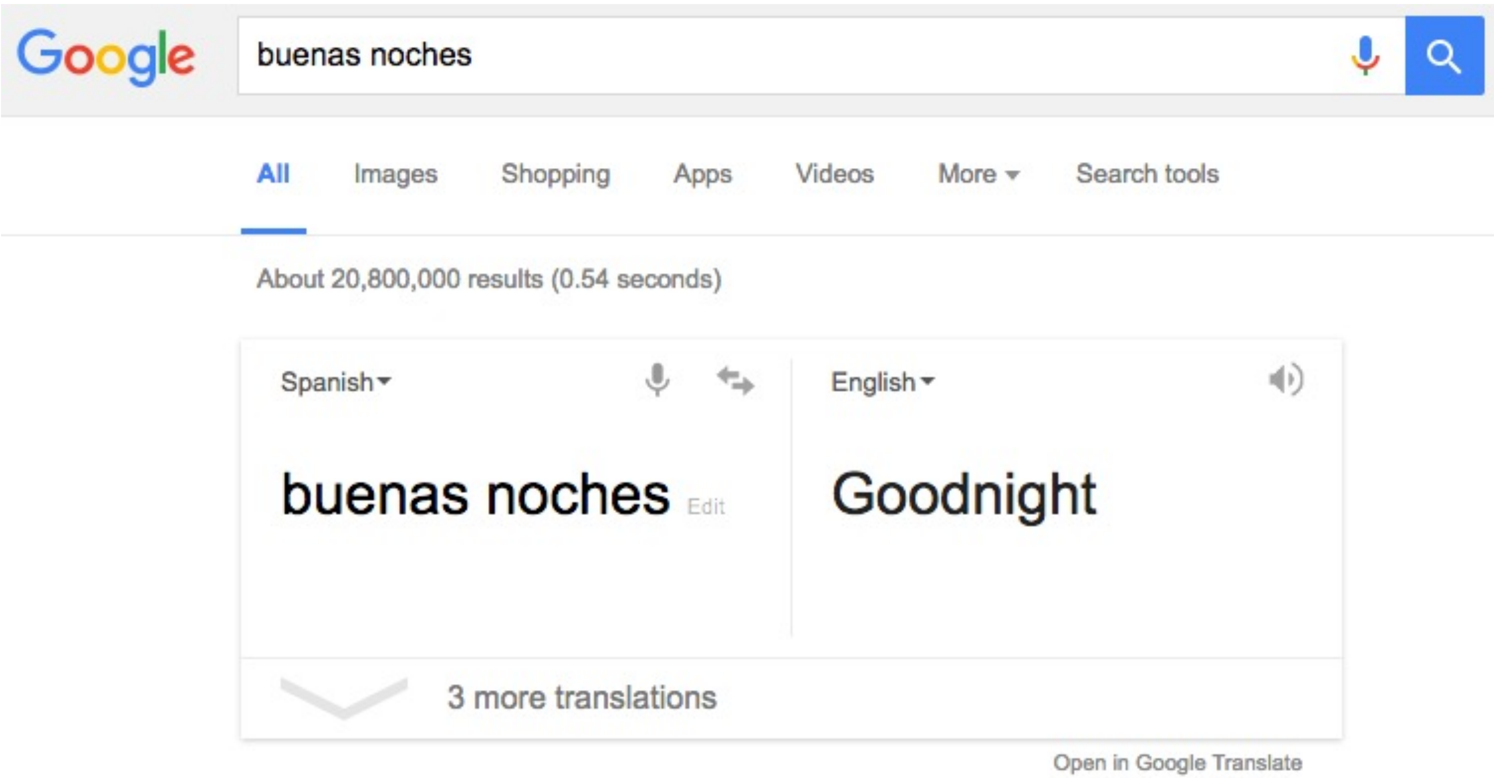
What is **Natural language processing** (NLP)

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

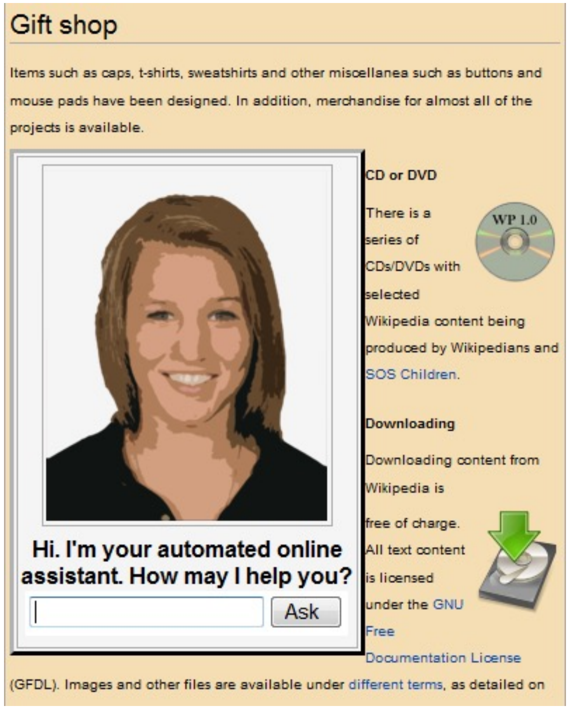
Credit: kaiwai Chang

Machine translation



Credit: kaiwai Chang

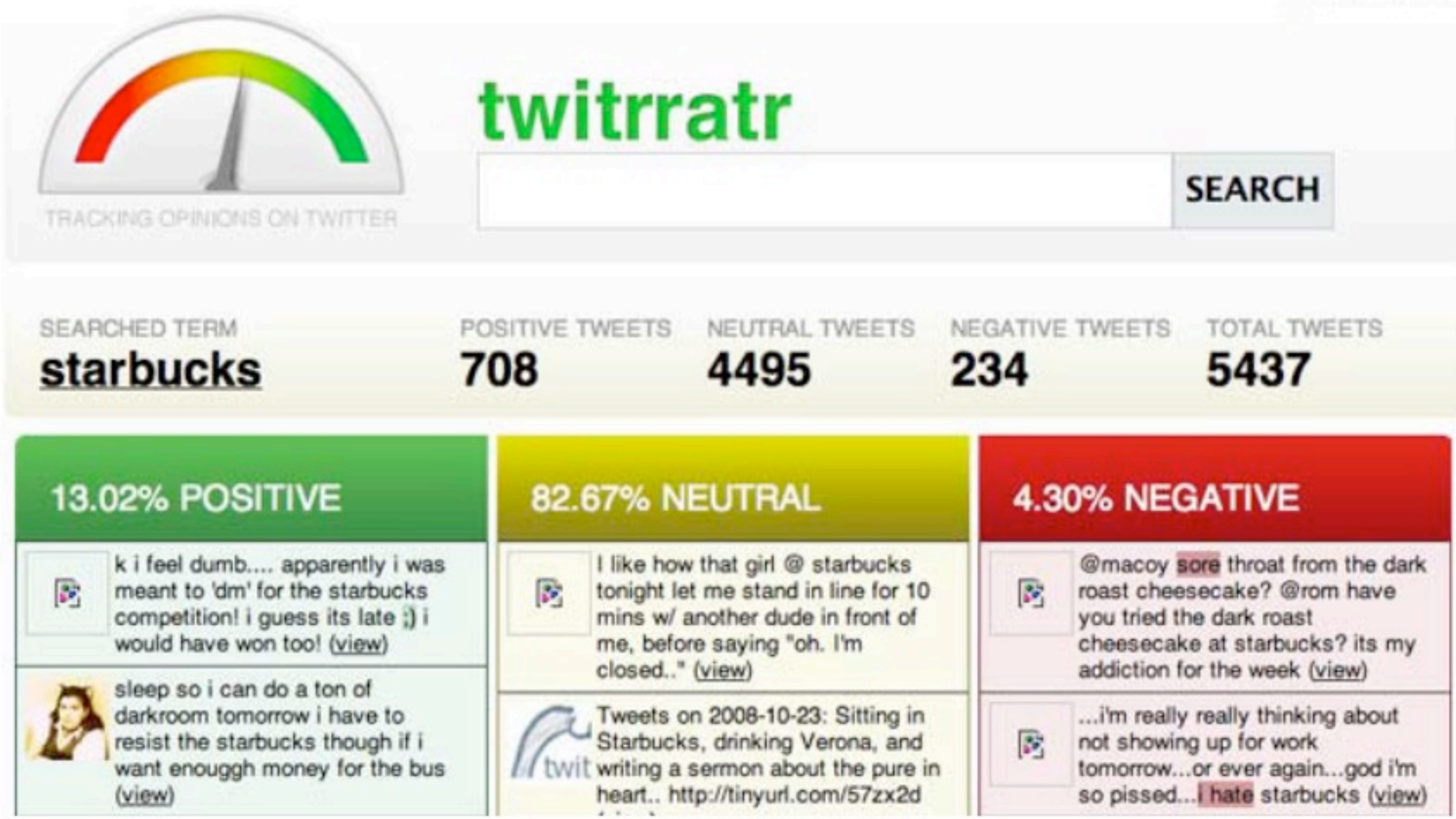
Dialog Systems



Natural language instruction

Credit: kaiwai Chang

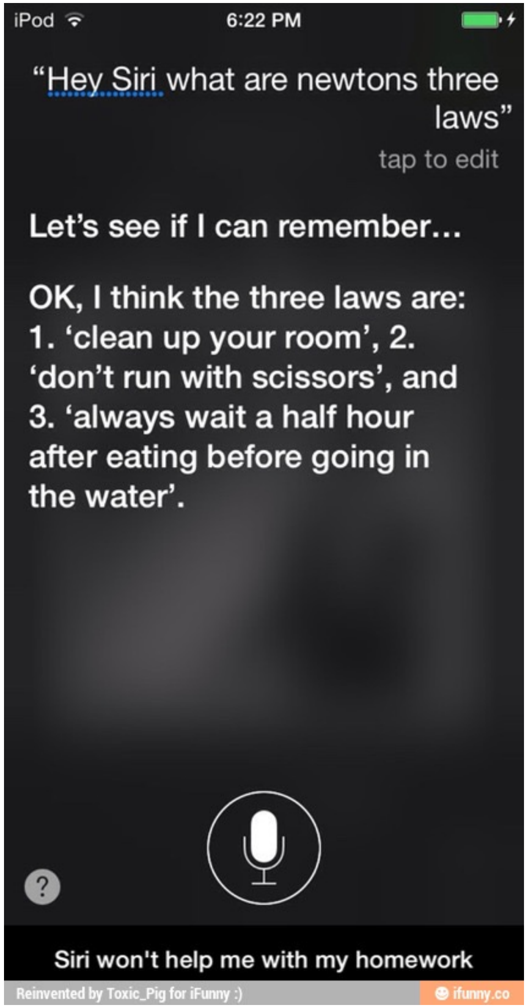
Sentiment/Opinion Analysis



Credit: kaiwai Chang

Question answering

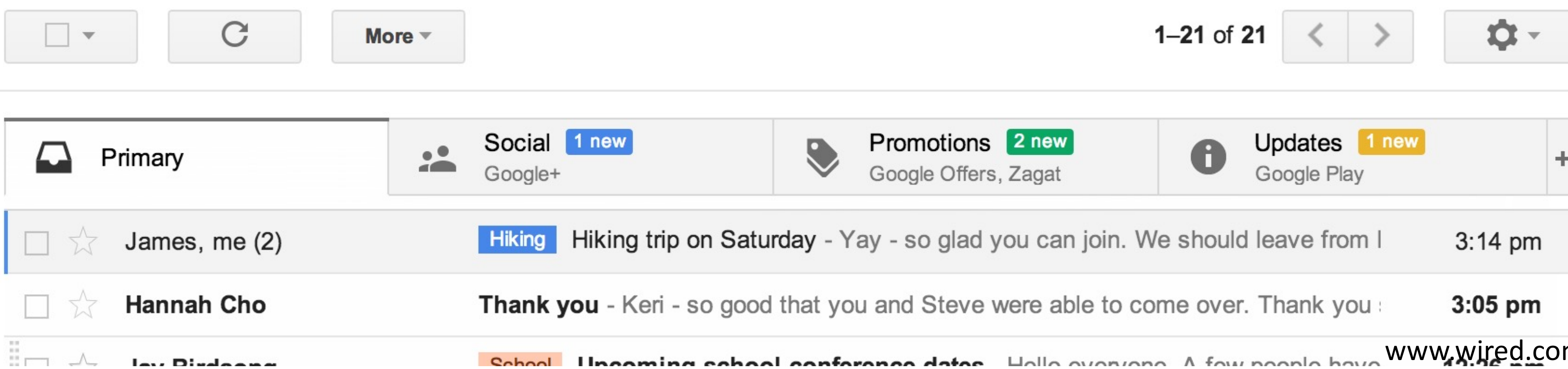
['Watson' computer wins at 'Jeopardy'](#)



credit: ifunny.com

Credit: kaiwai Chang

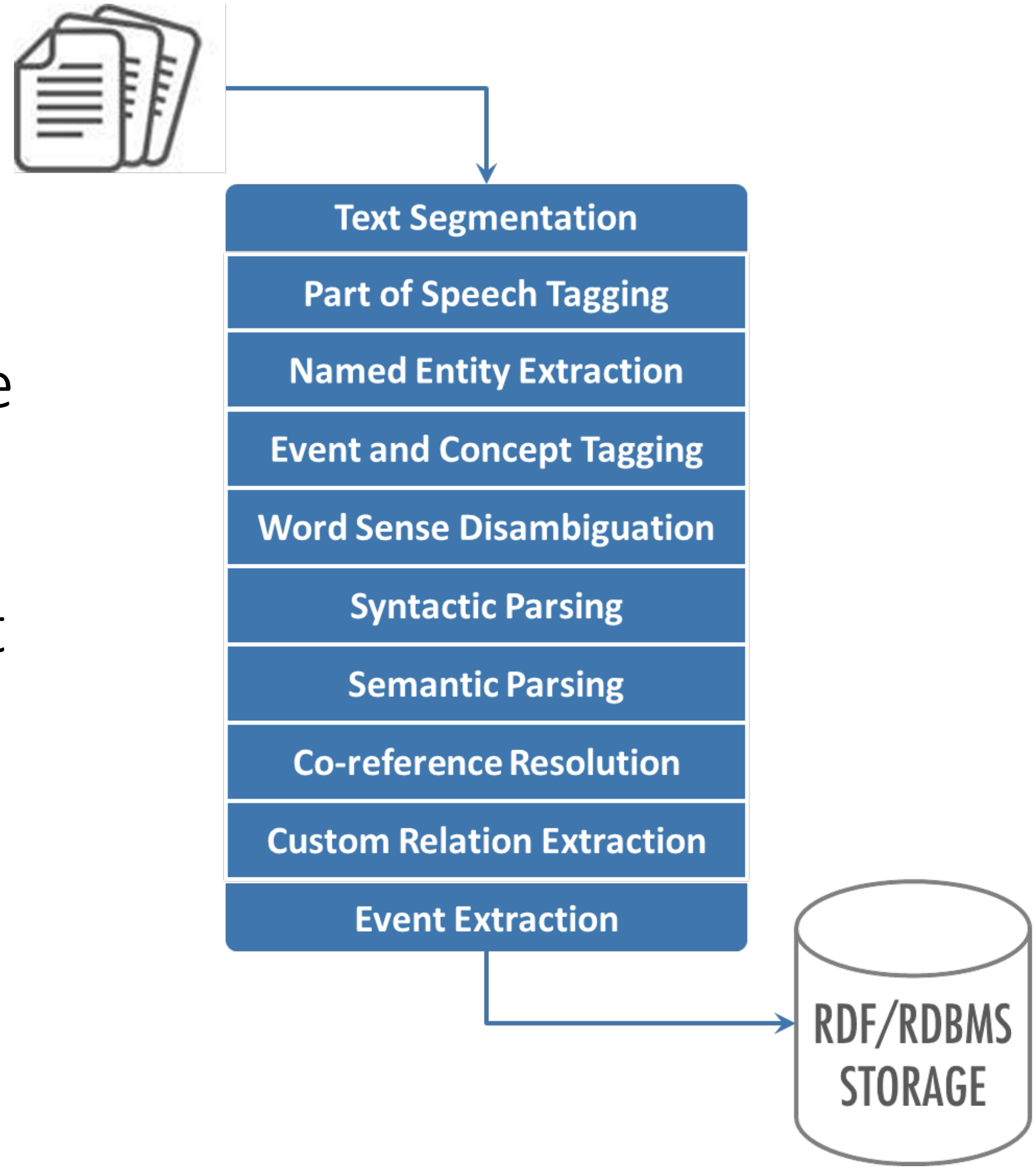
Text Classification



www.wired.com

Credit: kaiwai Chang

Classic NLP Pipeline Includes a set of Components for Understanding Text

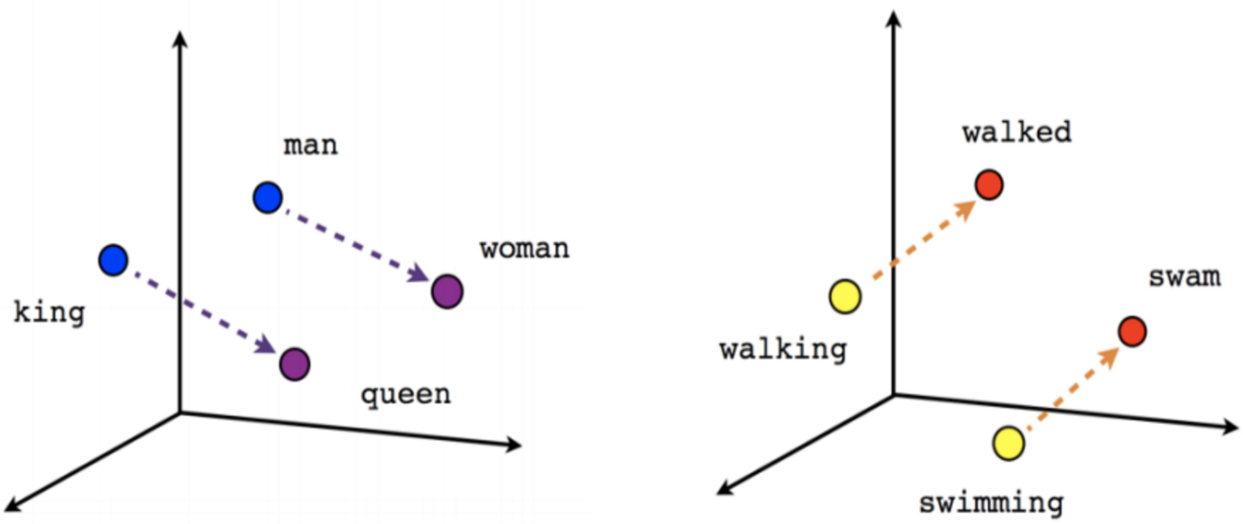


Credit: kaiwai Chang

Recent deep learning advances on natural language

- Before Deep NLP (Pre 2012)
  - Supervised predictors for each component
  - (BOW / LSI / Topic LDA)
- Word2Vec (2013-2016)
  - (GloVe/ FastText)
- Recurrent NN (2014-2016)
  - LSTM
  - Seq2Seq
- Attention / Self-Attention (2016 – now)
  - Attention
  - Transformer (self-attention, attention only)
  - BERT / XLNet/ GPT-2 / T5 …

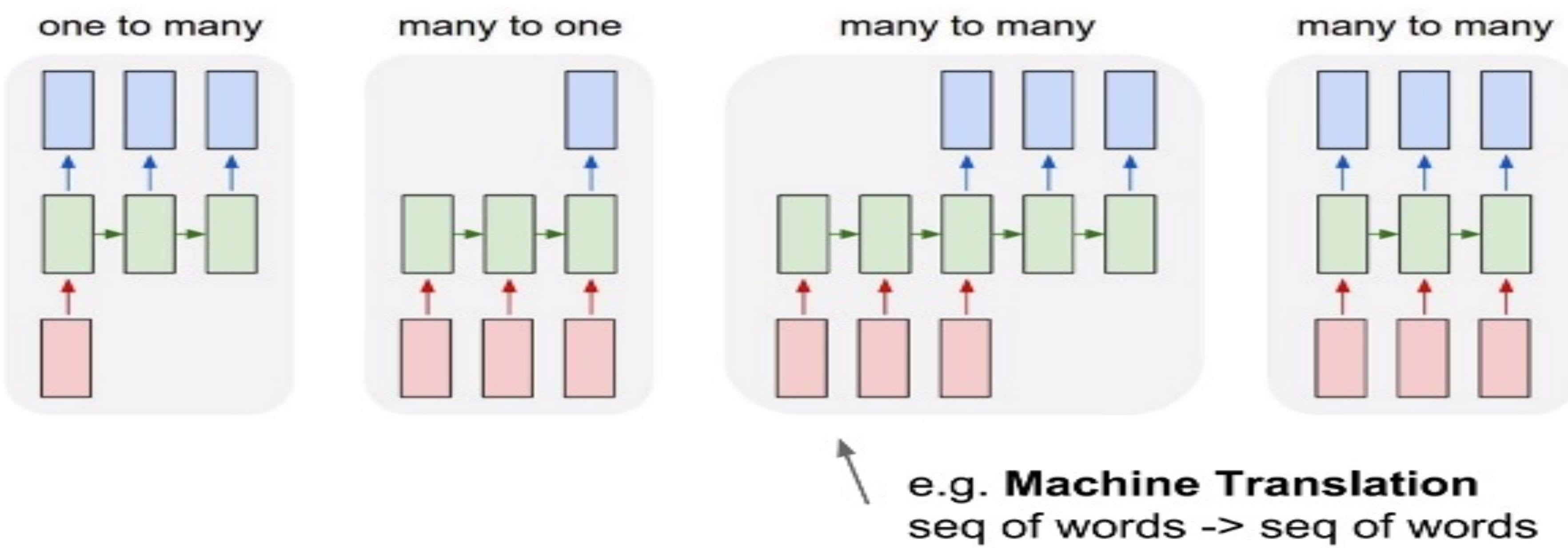
Distributional Word Embedding Vector: To Represent A Word in DNN

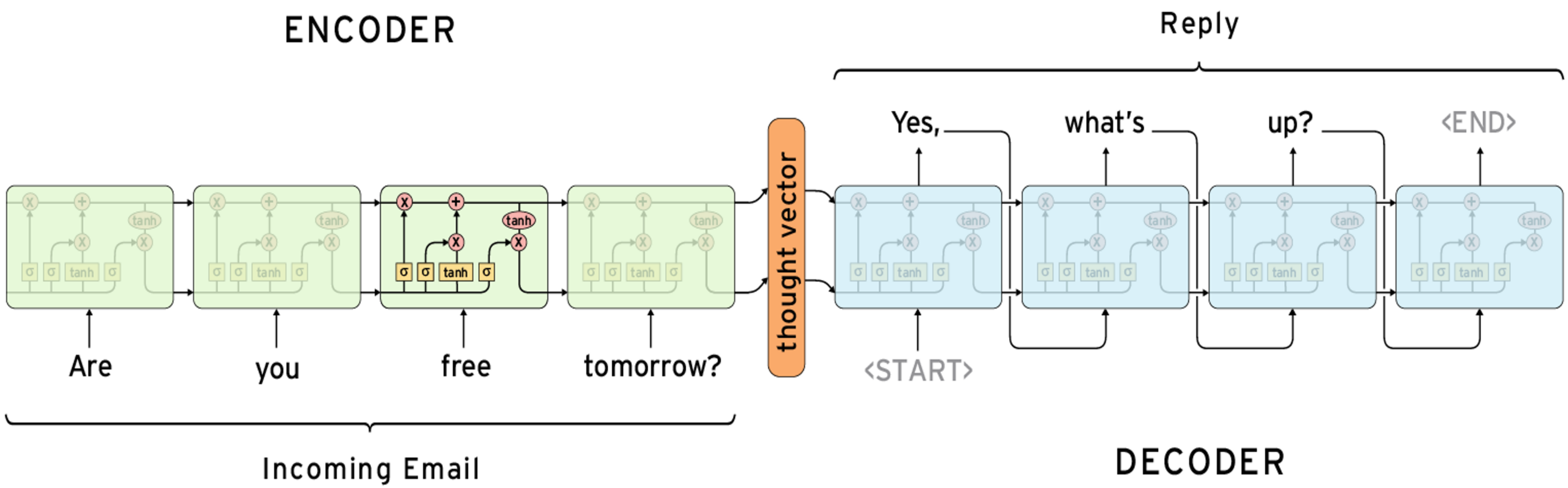


Male-Female

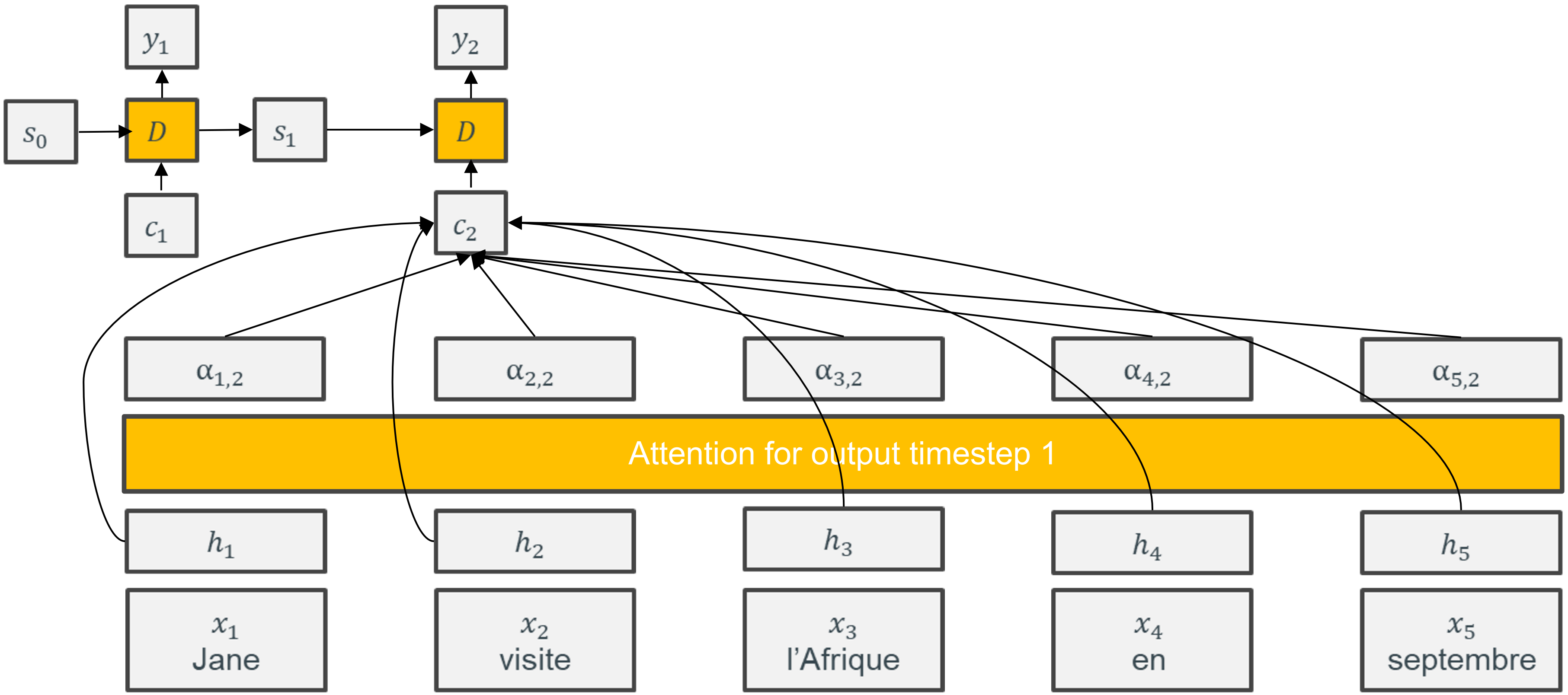
Verb tense

Recurrent Neural Networks (RNNs) can handle

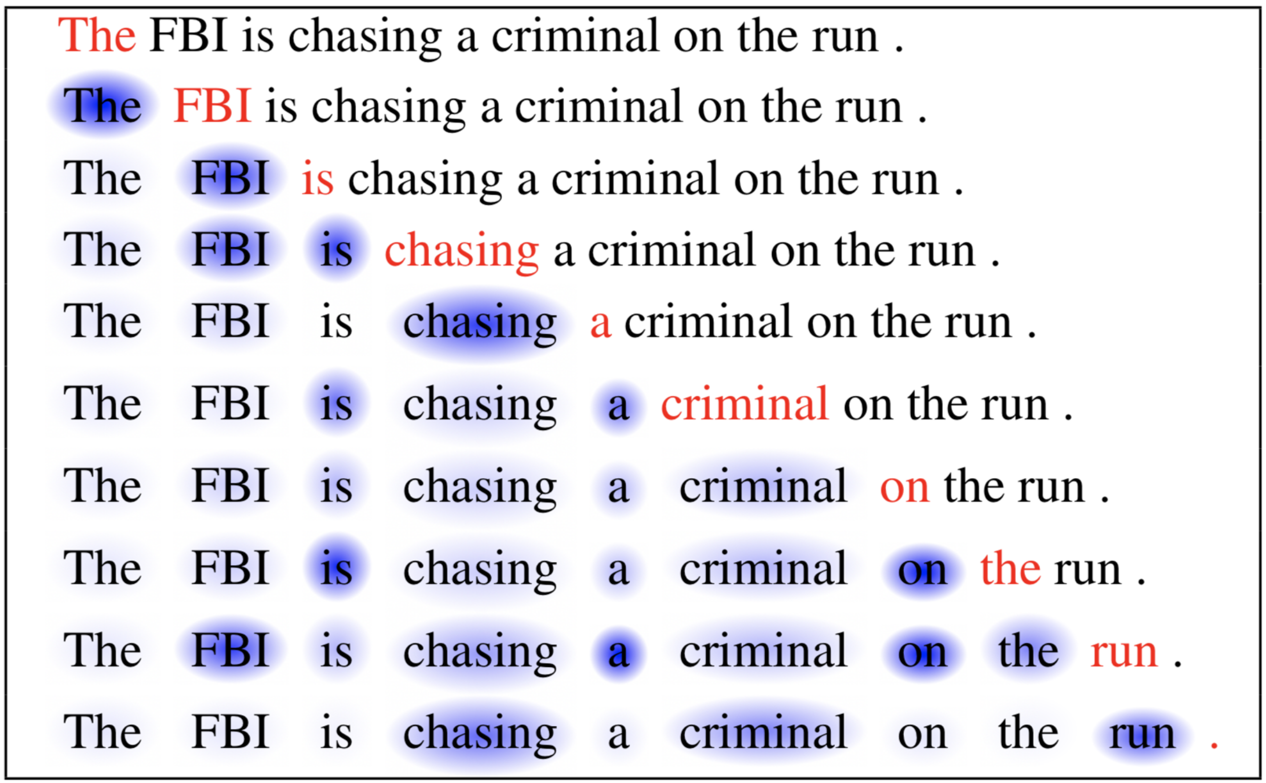




The attention module gives us a weight for each input.



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



Transformer: Exploiting Self Attentions

- A Google Brain model.
  - Variable-length input
  - Fixed-length output (but typically extended to a variable-length output)
  - **No recurrence**
  - Surprisingly not patented.
- Uses 3 kinds of attention
  - Encoder self-attention.
  - Decoder self-attention.
  - Encoder-decoder multi-head attention.

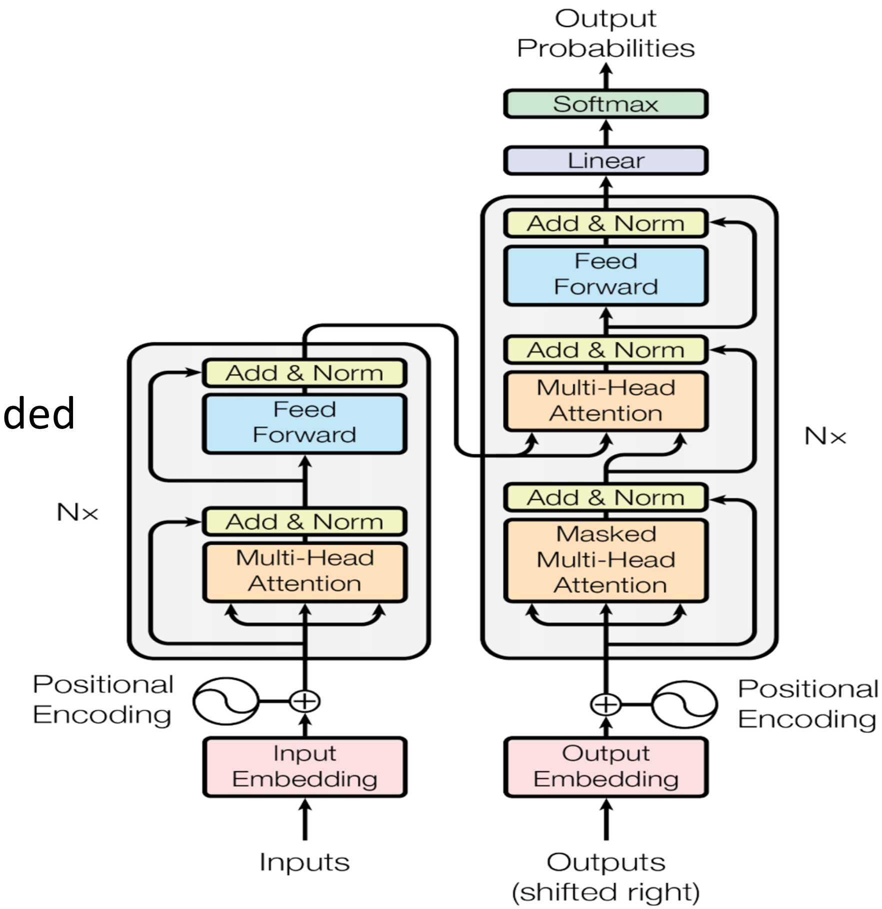
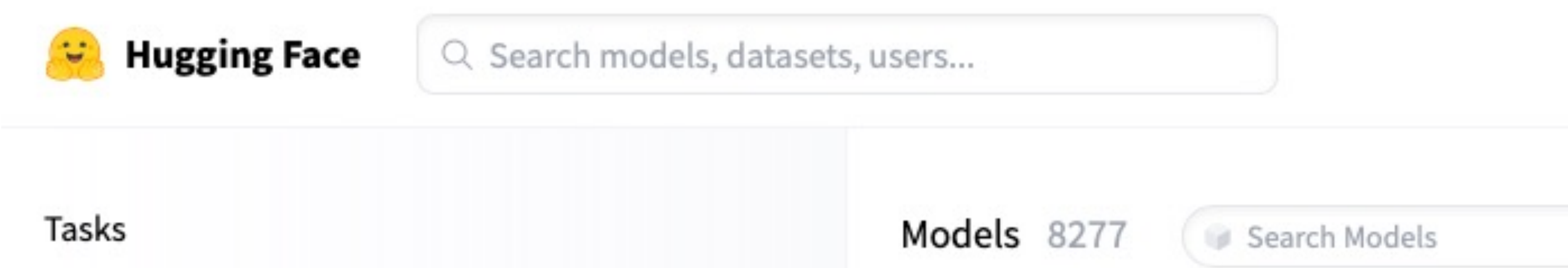
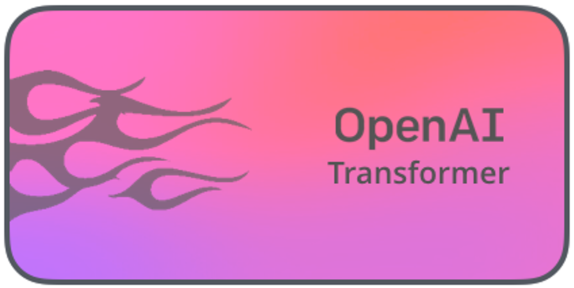


Figure 1: The Transformer - model architecture.

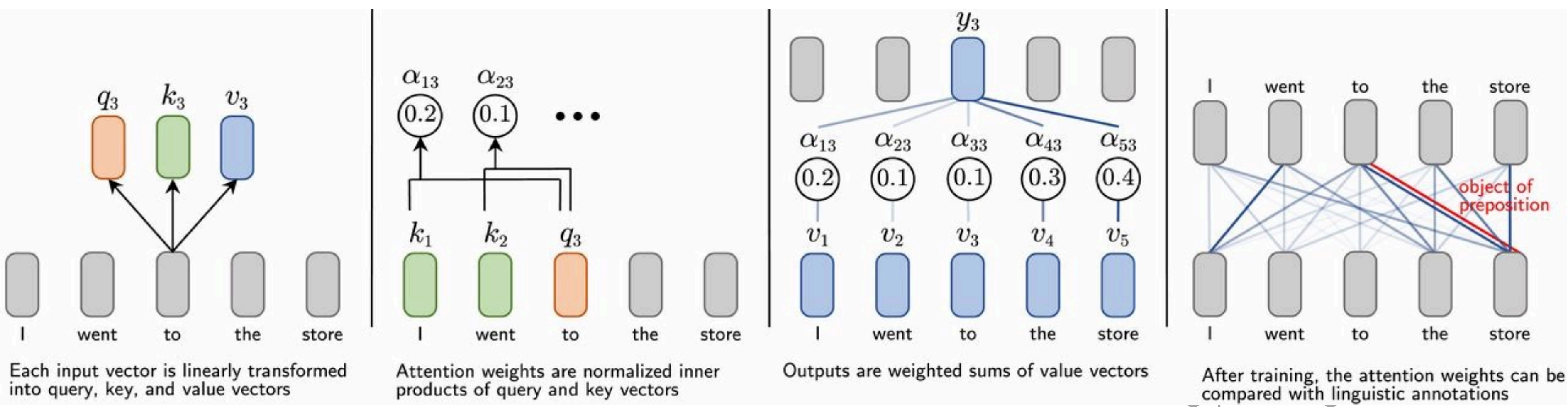




BERT: Bidirectional Encoder Representations from Transformers

Pre-trained transformer encoder for sentence embedding

Notable pre-trained NLP models



Background:

Adversarial Examples

Background: Adversarial Examples

x

+

Trained Deep learning Model

y: “panda”

x

$0.007 \times [noise]$

=

x’= x + r

+

Trained Deep learning Model

t: “gibbon”

x: original sample

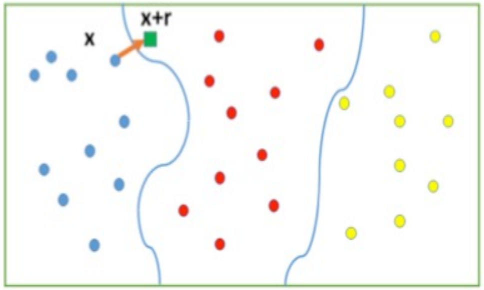


Image Space

x’ = x + r : adversarial sample

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

Background: Adversarial Examples

x

+

Trained Deep learning Model

y: “panda”

x

$0.007 \times [noise]$

=

x’= x + r

Trained Deep learning Model

t: “gibbon”

Many different variations of formulations to search for x’ from x,

Misclassification term | Distance term

$$minimize \ \|f(x') - t\| + \lambda * \Delta(x, x')$$

Misclassification term Distance term

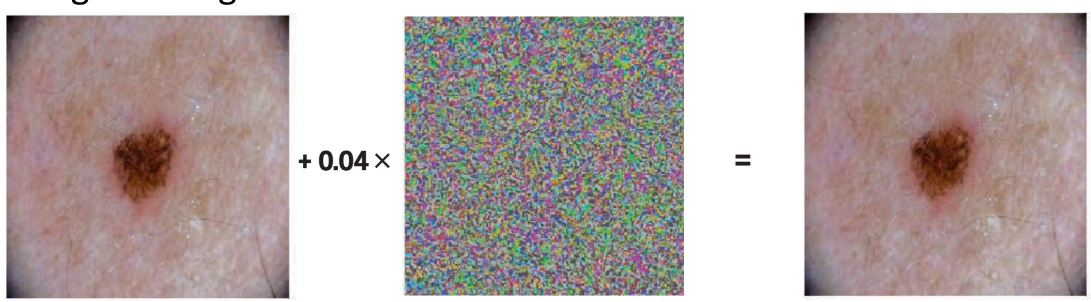
$$minimize \; \|f(x') - t\| + \lambda * \Delta(x, x')$$

Deep Learning Classifiers are Easily Fooled

Healthcare

Melanoma Diagnosis with Computer Vision

Original Image Perturbation Adversarial Example



Benign Malignant

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

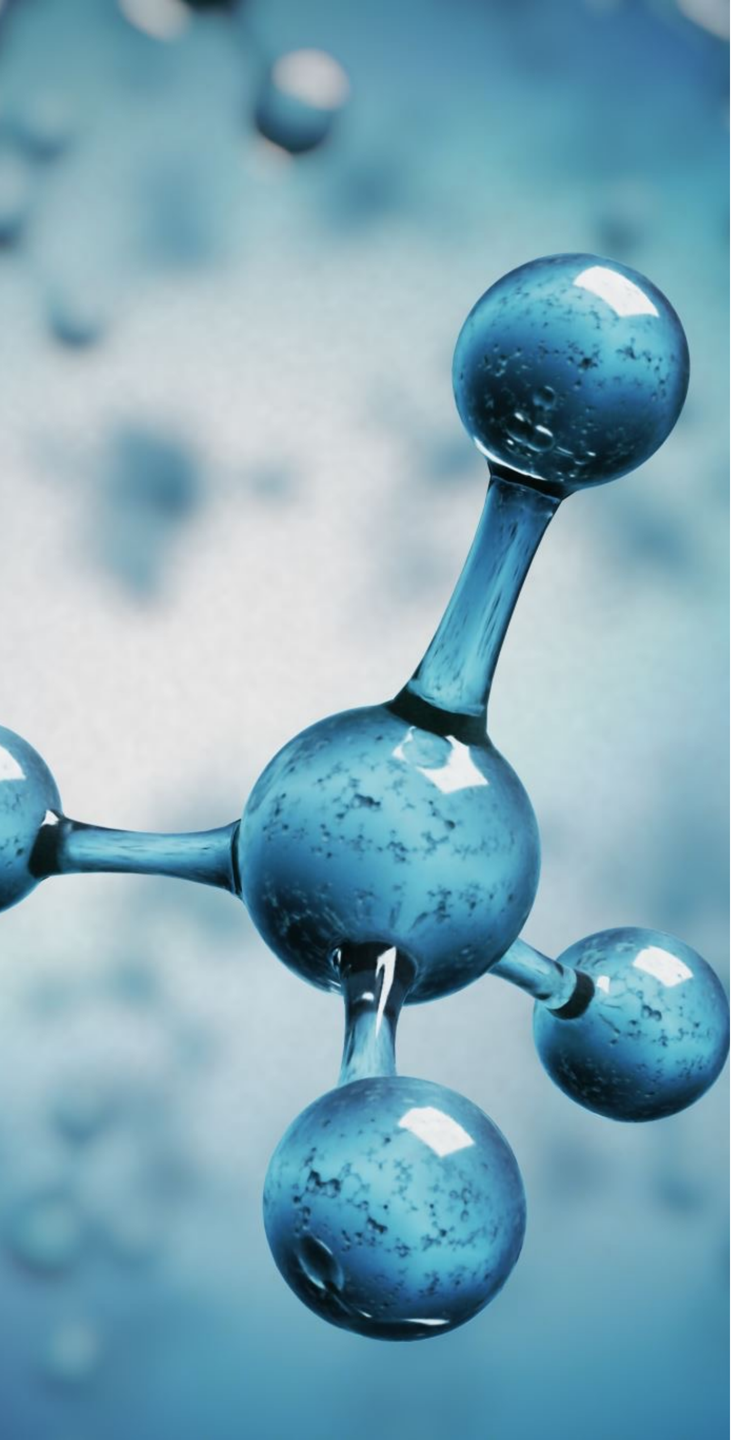
Michael K. Reiter
University of North Carolina
Chapel Hill, NC, USA
reiter@cs.unc.edu

ACM CCS 2016

Actual images

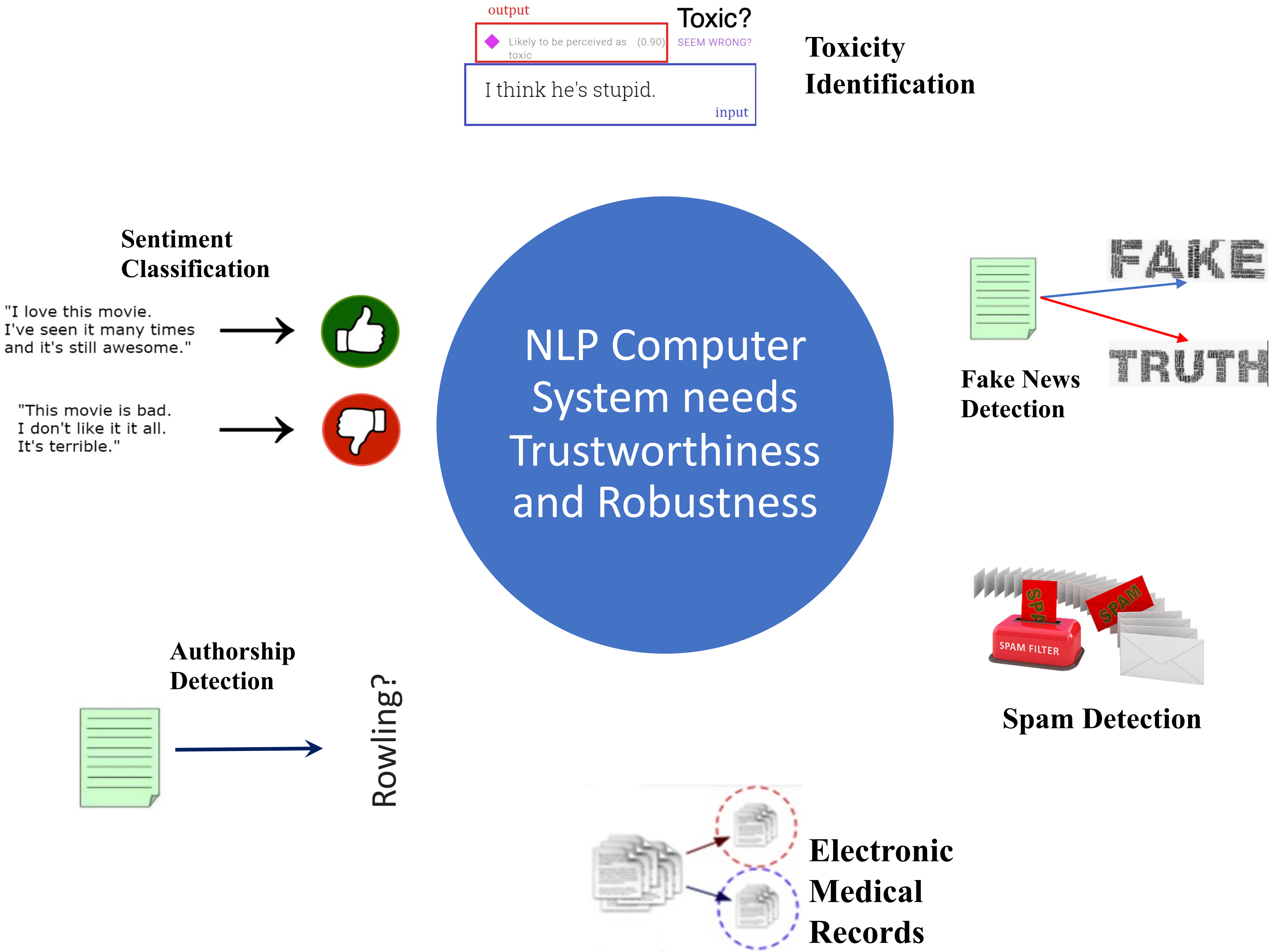
Recognized faces

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



Terminology

- Changes to inputs that fool the model are known as **adversarial examples** or **adversarial perturbations**
- A program that repeatedly generates adversarial examples for some model is known as an **adversarial attack**
- A model's resistance to adversarial examples is known as **robustness**


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

Misclassification term Distance term

$$minimize\ \|f(x') - t\| + \lambda * \Delta(x, x')$$

What are adversarial examples in NLP?

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

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 **moive** I've seen **snice** I was a small boy.

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)

Misclassification term   Distance term

$$minimize \; \|f(x') - t\| + \lambda * \Delta(x, x')$$

What are adversarial examples in NLP?

- **<u>Idea 2:</u>** examples that are indistinguishable *in meaning* to the original input (**semantics-preserving changes**)

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 movie I've seen since I was a **wee lad**.
Prediction: **<u>Positive ✓</u>**		Prediction: **<u>Negative ✗</u>**

AE NLP literature is messy (chaotic)

1. Many generate examples are bad
2. No standard library
3. No clear benchmarking insights
4. No clear benefits

Our Solution:

TextAttack to Rescue

1. Many generate examples are bad

AE NLP literature is messy (chaotic)

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 our transformation only produces "valid" examples?

- **<u>Idea 1:</u>** what is the cosine similarity between the sentence embeddings of $\mathbf{x}$ and $\mathbf{x}_{\mathbf{adv}}$**?**
  - (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 $\mathbf{x}_{\mathbf{adv}}$.

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

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

Our Analysis paper: Reevaluating Adversarial Examples in Natural Language
• 2020 EMNLP Findings

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

Standardize Constraints Enables Better/ Truthful Comparison

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

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

Difficult to utilize attacks and attack components for improving models

- Attack implementations are almost never model-agnostic
- Adversarial training code is usually unreleased or non-existent
- Data augmentation not nearly as commonplace as in images

Generating NLP adversarial examples

Four Components Framework:

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

$$\underbrace{minimize\ \|f(x') - t\|}_{\text{Goal Function term}} + \underbrace{\lambda * \Delta(x, x')}_{\text{Constraints' term}}$$

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

•2020 EMNLP Demo

Transformation: Word Substitution centered

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

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

How can we use transformations and constraints to attack a NLP model?

- We need two more things:
  - 1. A way to **search** the space of transformations for a valid, successful adversarial example.
  - 2. A way to know whether an example successfully fools the model.

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

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

Goal Function term   Constraints' term

$$minimize\ \|f(x') - t\| + \lambda * \Delta(x, x')$$

The TextAttack Framework

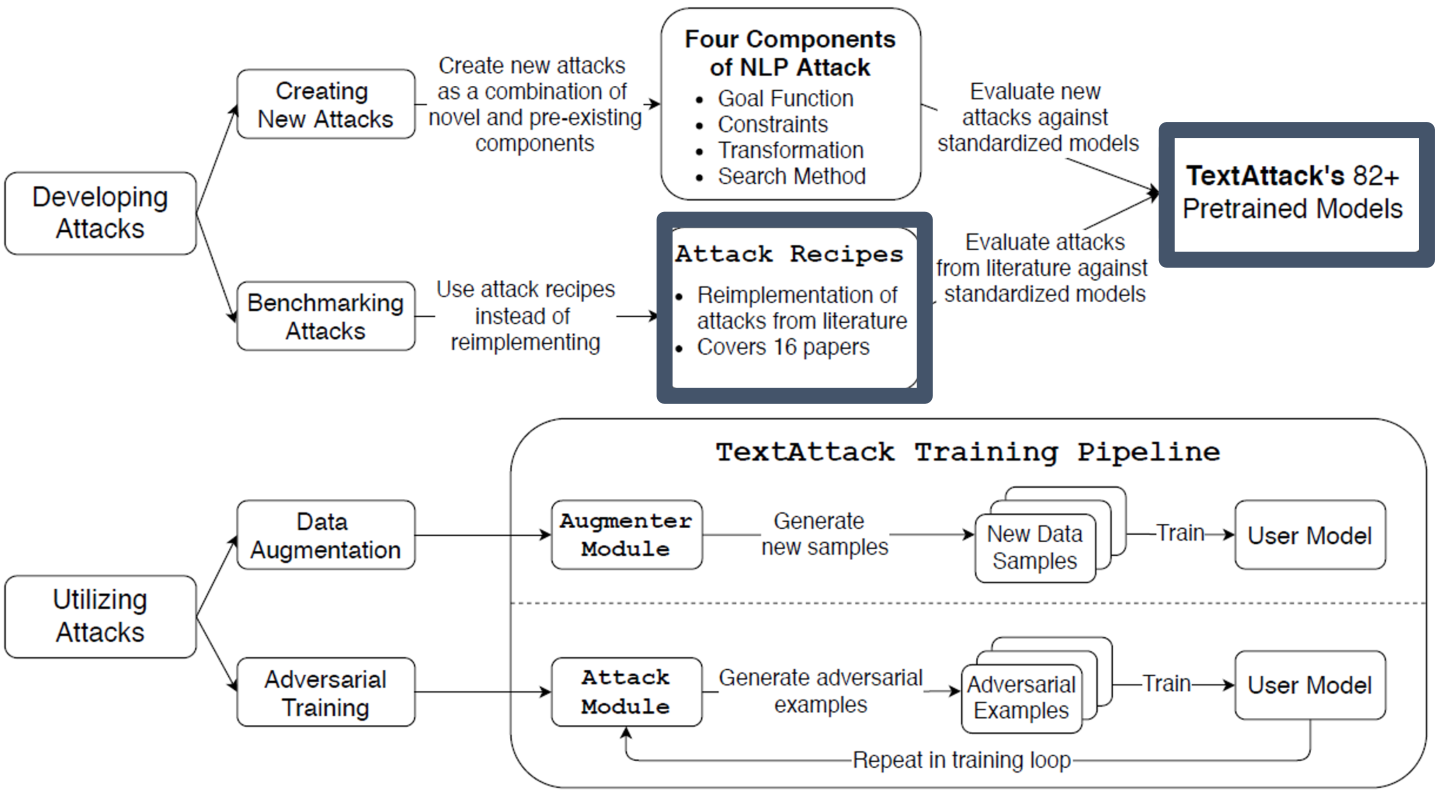
NLP attacks can be constructed from four components:

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

Goal Function term   Constraints' term

$$minimize \; ||f(x') - t|| + \lambda * \Delta(x, x')$$

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]

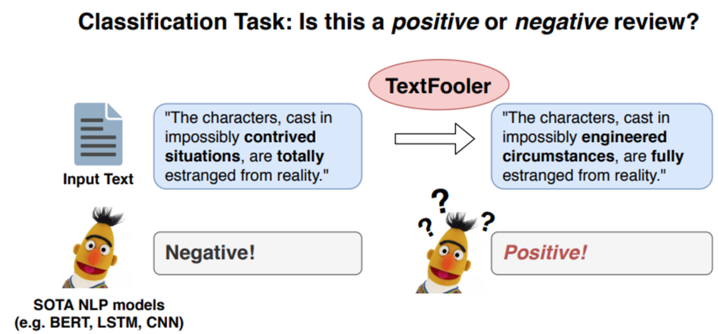


Figure 1: Our model TextFooler slightly change the input text but completely altered the prediction result.

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.
    CANDIDATES $\leftarrow$ POSFilter(CANDIDATES)
    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 $Y_k \neq Y$
        $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\mathrm{argmax}} \ \mathrm{Sim}(X, X'_{w_j \rightarrow c})$
        $X_{\mathrm{adv}} \leftarrow$ Replace w_j with c^* in X_{adv}
        return X_{adv}
    else if $P_{Y_k}(X_{\mathrm{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\min} P_k$ **then**
        $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\mathrm{argmin}} P_k$
        $X_{\mathrm{adv}} \leftarrow$ 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)

Search Algorithm: Greedy with Word Importance Ranking

Transformation: Counter-fitted embedding word swap

Constraint #3: Cosine similarity of sentence embeddings

Goal Function: Untargeted attack for classification

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
```

Constraint #1: Cosine similarity of word embeddings

Constraint #2: Consistent part-of-speech

Four Components Standardized 18 Attacks:

	Alzantot et al. (2018)	Jin et al. (2019)
Goal Function	UntargetedClassification	UntargetedClassification
Search Method	GeneticAlgorithmWordSwap	GreedyWordSwapWordImportanceRanking
Transformation	WordSwapEmbedding(embedding='cf')	WordSwapEmbedding(embedding='cf')
Constraints	• WordsPerturbedPercentage(max_perc=20) • WordEmbeddingDistance(max_mse=0.5) • GoogleLanguageModel(n_per_index=4)	• WordEmbeddingDistance(min_cos_sim=0.5) • PartOfSpeech(verb_noun_swap=True) • UniversalSentenceEncoder(metric='angular', thresh=0.904458599)

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

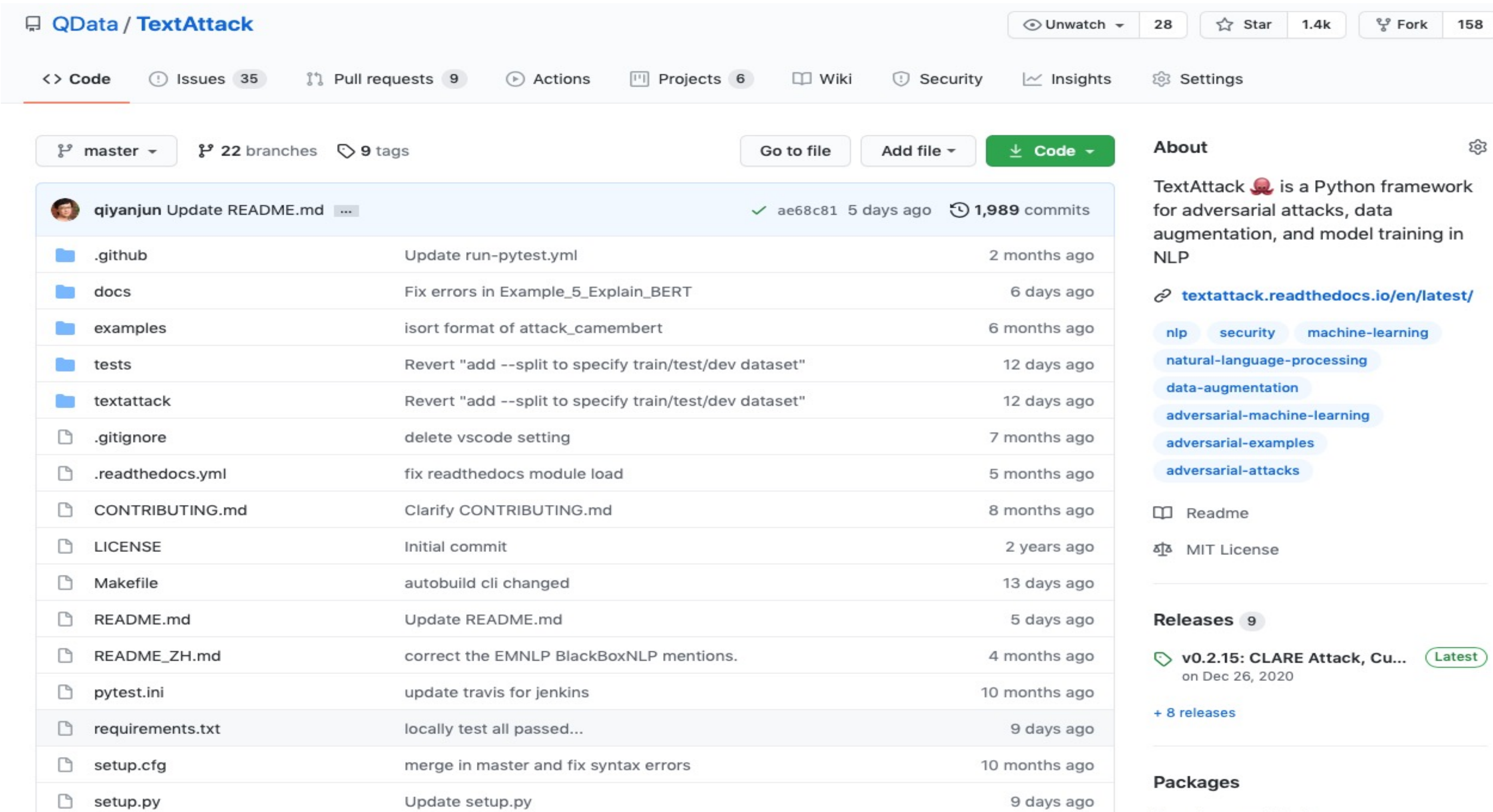
Trained on all GLUE tasks

Installing TextAttack

Github PyTest passing | pypi package 0.2.12

pip install textattack

https://github.com/QData/TextAttack



AE NLP literature is messy (chaotic)

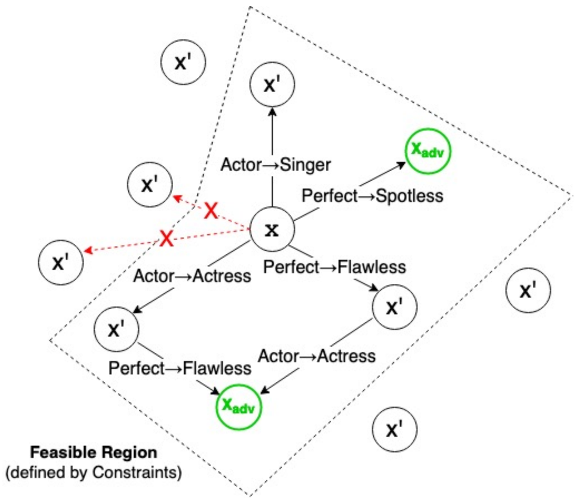
3. No clear benchmarking insights

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]

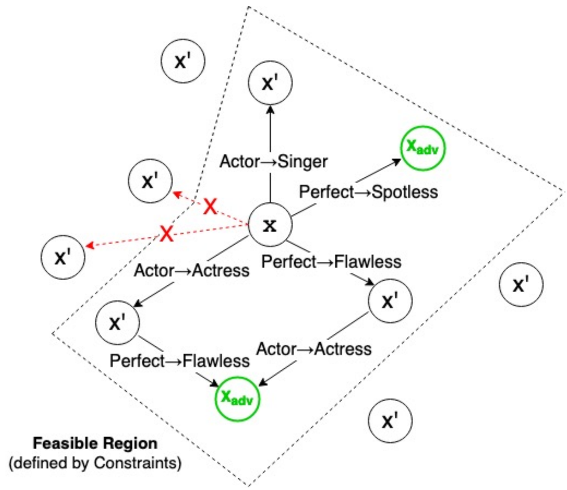
Search Algorithm: A way to **search** the space of transformations for a valid, successful adversarial example.

Search Space

Search space defined by transformation and constraints

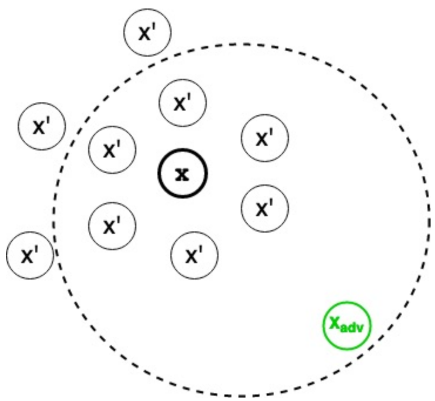
Let $T(x)$ be our transformation and $C_i(x)$ be a constraint,

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

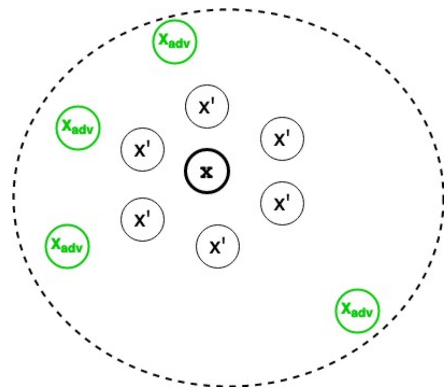


Search Space

How search space is defined can affect performance of the search algorithm



Strict constraints



Lax constraints

Search Algorithms from Literature

A lot of works have proposed novel search algorithms.

Proposed search algorithms:

- **Greedy:** (Kuleshov et al. 2018)
- **Beam Search:** (Ebrahimi et al., 2017)
- **Greedy with Word Importance Ranking:** (Gao et al., 2018), (Jin et al., 2019), (Ren et al., 2019)
- **Genetic Algorithm:** (Alzantot et al., 2018),
- **Particle Swarm Optimization:** (Zang et al., 2020)
- **MCMC Sampling:** (Zhang et al., 2019)

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

Performance across different search methods

Model	Dataset	Search Method	Lax Constraint: $sim = 0.5$			Strict Constraint: $sim = 0.9$		
			A.S. %	P.W. %	Queries	A.S. %	P.W. %	Queries
BERT	Yelp	Word Importance Ranking (UNK)	99.8	8.59	393	25.7	10.69	219
		Word Importance Ranking (DEL)	99.7	9.16	423	26.1	10.73	220
		Word Importance Ranking (RAND)	99.7	16.43	610	23.5	12.58	94
		Greedy (b=1)	99.8	5.02	9,813	30.3	7.59	1,984
		Beam Search (b=4)	100.0	4.92	30,417	31.1	7.59	7,297
		Beam Search (b=8)	**100.0**	4.89	57,984	**31.3**	7.59	14,329
		Genetic Algorithm	99.6	9.83	7,173	21.5	9.52	12,655
	MR	Word Importance Ranking (UNK)	99.2	15.58	116	30.2	14.5	34
		Word Importance Ranking (DEL)	98.8	15.00	113	30.8	15.04	34
		Word Importance Ranking (RAND)	99.1	21.00	132	29.1	16.00	16
		Greedy (b=1)	99.3	11.86	639	31.1	11.51	49
		Beam Search (b=4)	99.7	11.68	1,411	32.1	11.64	141
		Beam Search (b=8)	**99.7**	11.61	2,432	**32.3**	11.67	261
		Genetic Algorithm	99.4	14.93	1,611	31.4	13.47	2,870
	SNLI	Word Importance Ranking (UNK)	100.0	7.05	66	35.5	10.45	34
		Word Importance Ranking (DEL)	100.0	7.49	68	35.4	10.52	34
		Word Importance Ranking (RAND)	99.9	13.60	89	33.1	12.22	13
		Greedy (b=1)	100.0	6.08	473	38.3	7.94	37
		Beam Search (b=4)	100.0	6.02	662	39.7	7.97	95
		Beam Search (b=8)	**100.0**	6.02	918	**40.1**	8.09	173
		Genetic Algorithm	100.0	7.05	996	39.1	9.44	2,332

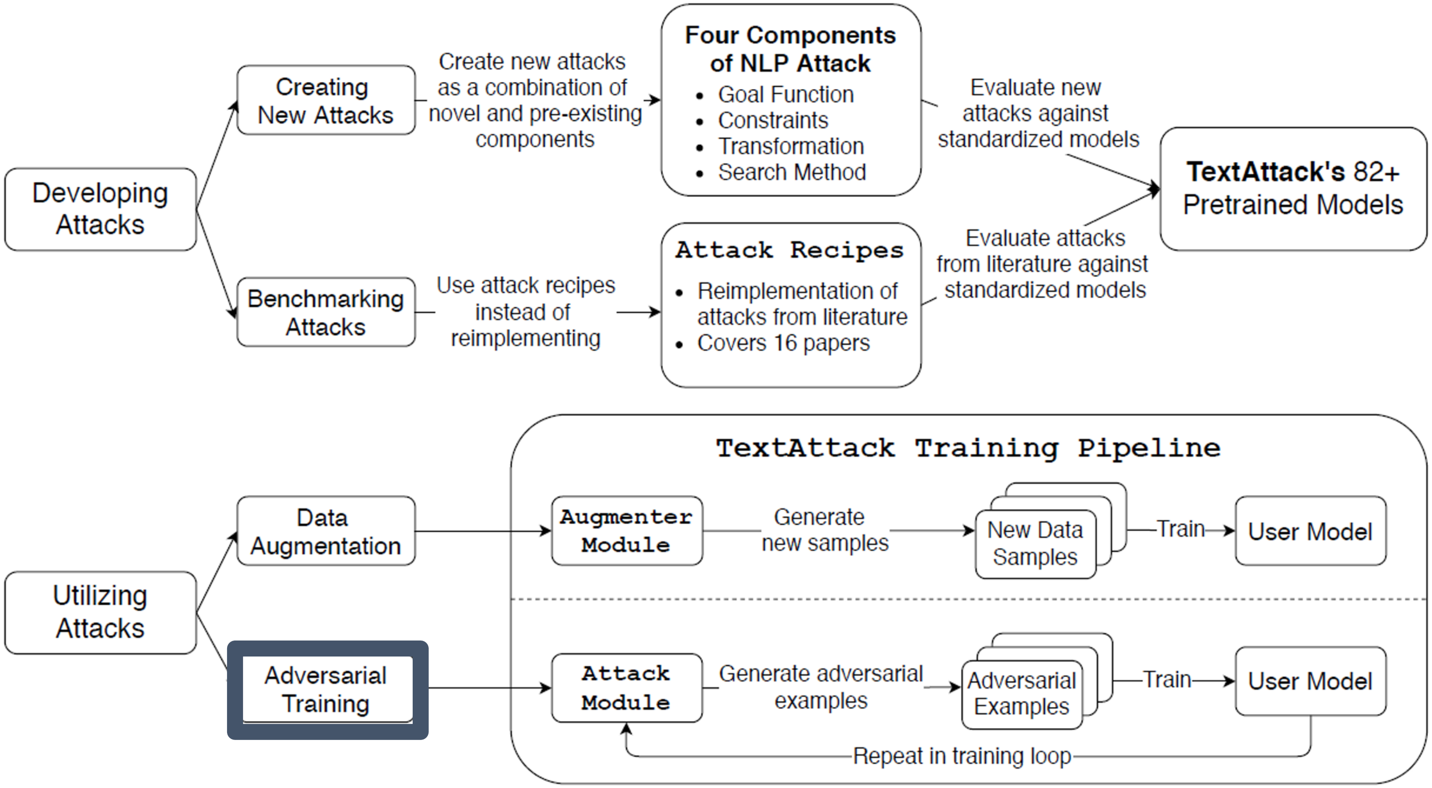
Benchmarking Insights

- Optimal method 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%.
- 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%.

AE NLP literature is messy (chaotic)

4. No clear benefits

Adversarial Training



Adversarial Training for Robustness

Goodfellow et al. (2015):

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

Madry et al. (2017):

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

Kannan et al. (2018):

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

IMDB-BERT Results

	IMDB Test Acc	Yelp Test Acc	Counterfactual Acc
No adv. training	93.97	92.86	92.84
SSMBA	93.94	92.52	92.48
Backtranslation	93.97	92.62	92.58
Textfooler-Mod	**94.49**	**93.29**	**93.23**
BAE-Mod	93.05	91.61	91.35

Our Analysis paper: Adversarial Training for Robust NLP Models

•2021 To Submit

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

What can I do with TextAttack?

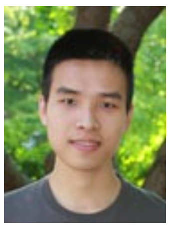
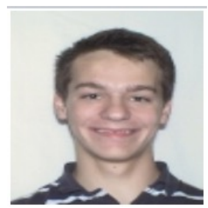
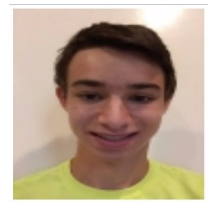
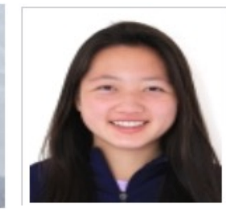
- build an NLP attacks from a library of components
- run those attacks on models & datasets (yours our ours)
- visualize attack results using the command line, Visdom, W&B, etc.
- or, use the infrastructure of TextAttack to develop and benchmark your own NLP attacks
- or, use the components from TextAttack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

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 the limitations of NLP models and 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%

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

http://trustworthymachinelearning.org

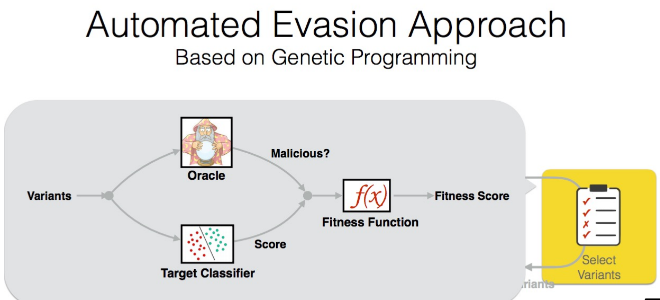
1. To Fool / Evade Learned Models

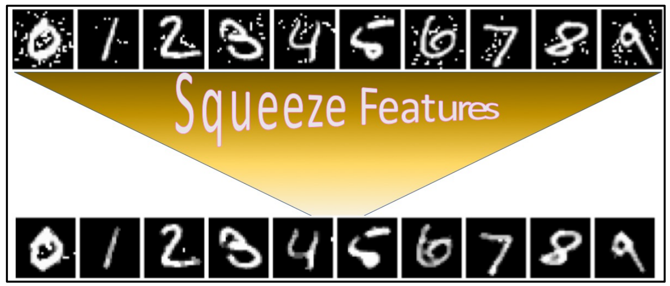
2. To Detect fooling/ Evasion

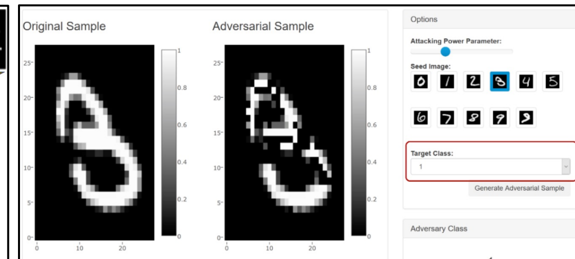
3. To Defend Against Evasion

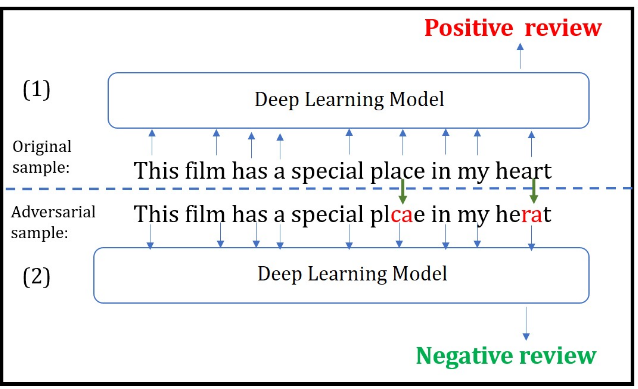
4. To Visualize and Benchmarking

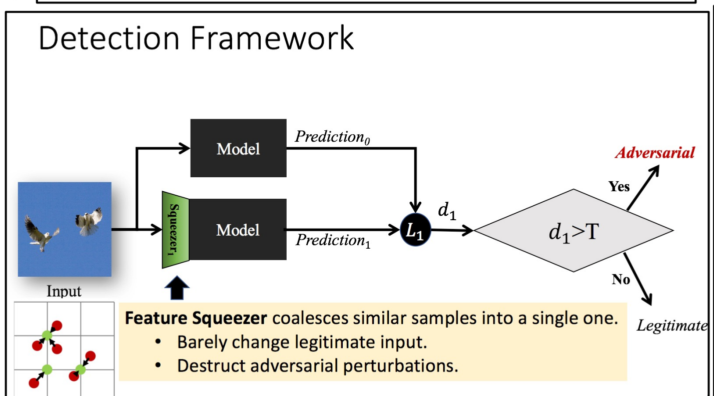
5. To Understand Theoretically

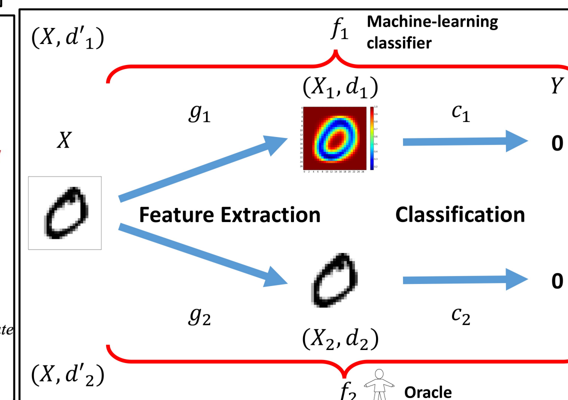






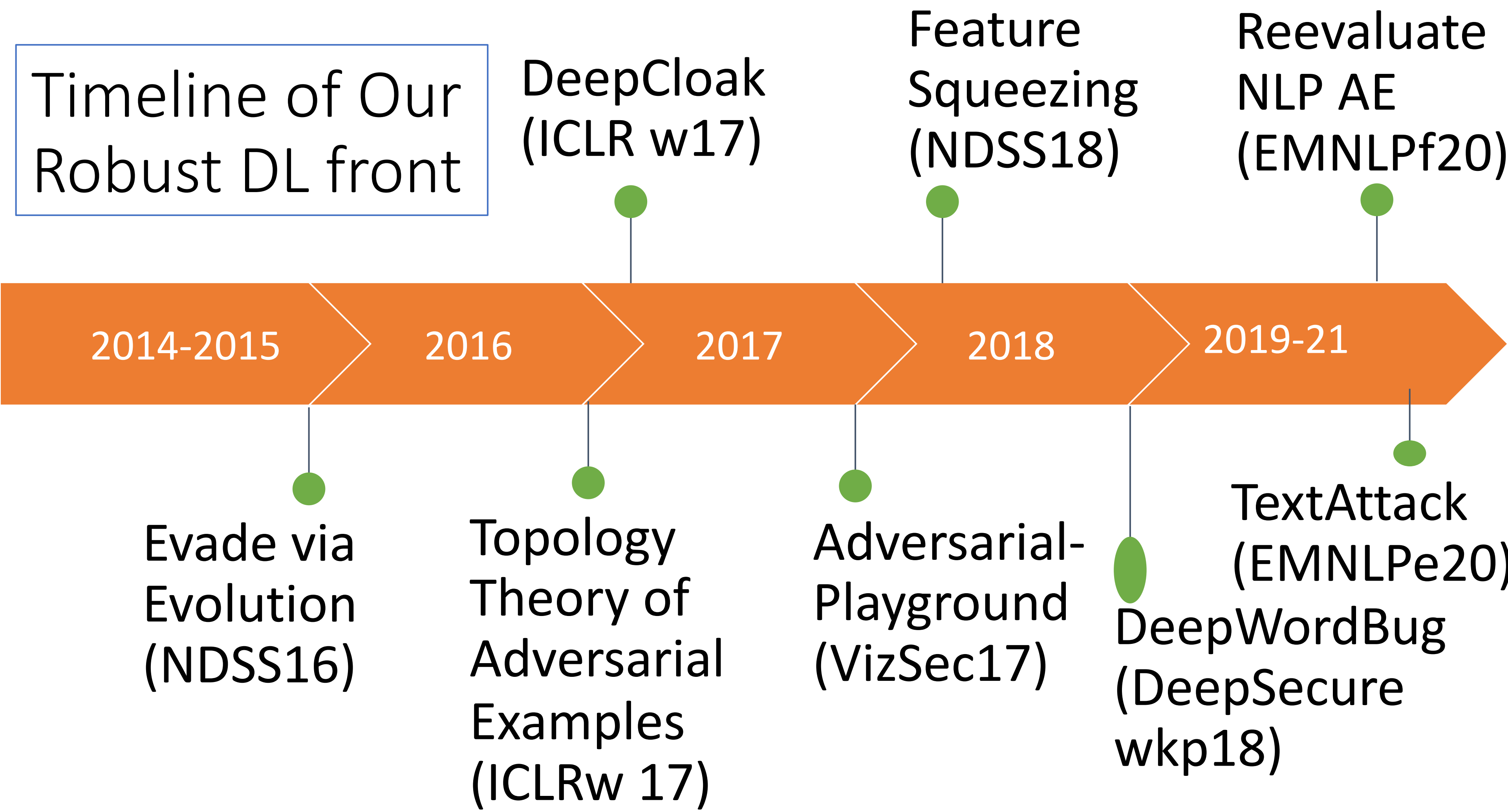






http://trustworthymachinelearning.org/

Timeline of Our Robust DL front



Thank you