



UNIVERSITY
of VIRGINIA

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/>



UNIVERSITY
of VIRGINIA

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

Machine translation

A screenshot of a Google search results page. The search query "buenas noches" is entered in the search bar. Below the search bar, the "All" tab is selected, followed by "Images", "Shopping", "Apps", "Videos", "More", and "Search tools". A message indicates "About 20,800,000 results (0.54 seconds)". The main content area features a machine translation card. On the left, under "Spanish", the phrase "buenas noches" is displayed with an "Edit" link. On the right, under "English", the translation "Goodnight" is shown. There are also microphone and speaker icons. Below the card, a button says "3 more translations". At the bottom of the card, there is a link "Open in Google Translate".

Dialog Systems

Gift shop

Items such as caps, t-shirts, sweatshirts and other miscellanea such as buttons and mouse pads have been designed. In addition, merchandise for almost all of the projects is available.



CD or DVD
There is a series of CDs/DVDs with selected Wikipedia content being produced by Wikipedians and SOS Children.

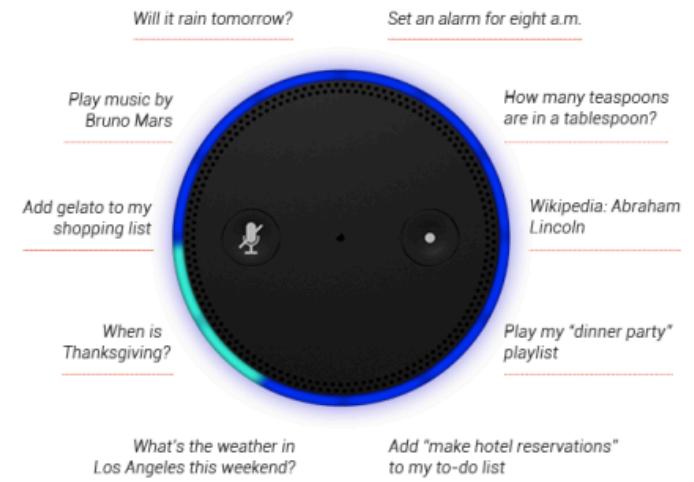
Downloading
Downloading content from Wikipedia is free of charge.
All text content is licensed under the [GNU Free Documentation License](#)

Hi. I'm your automated online assistant. How may I help you?

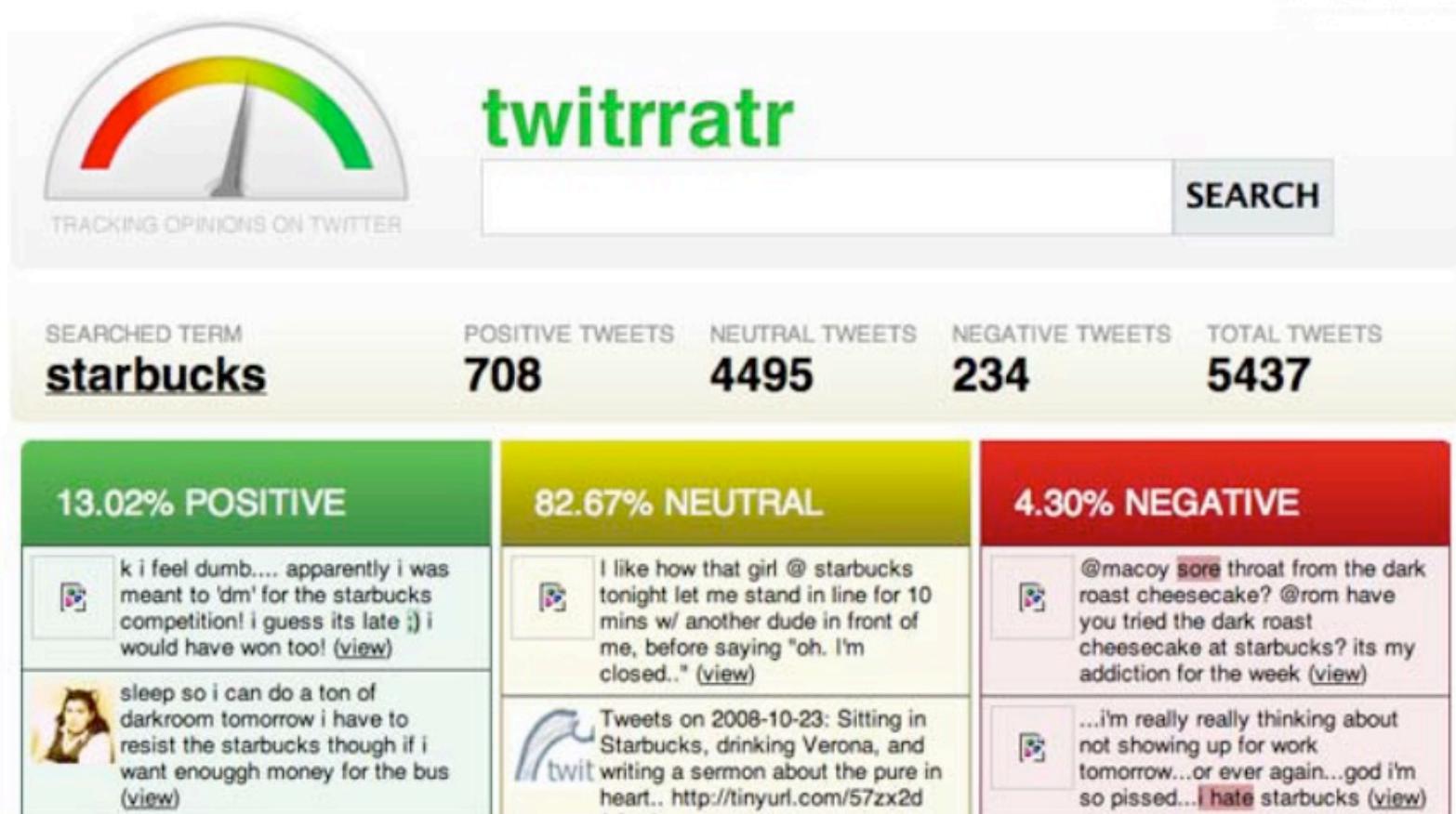
Ask

(GFDL). Images and other files are available under different terms, as detailed on

Natural language instruction



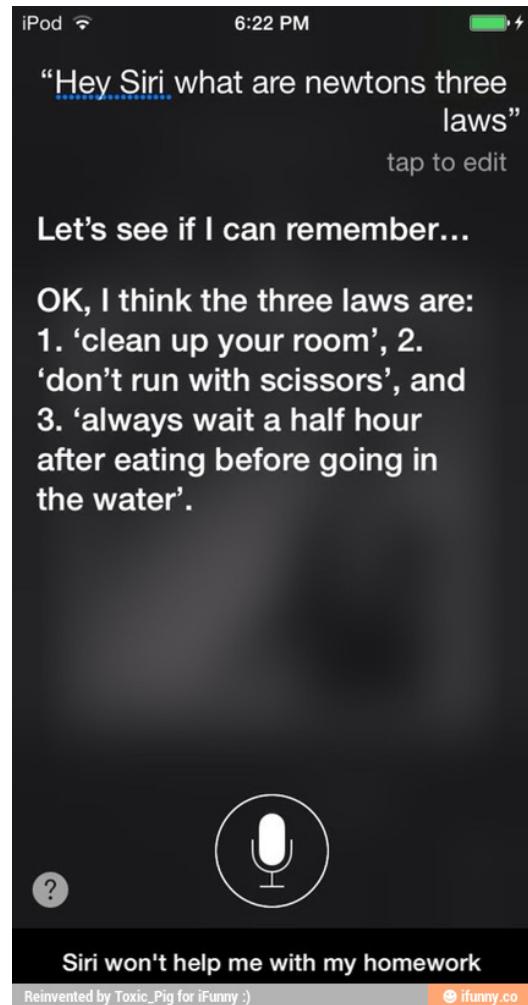
Sentiment/Opinion Analysis



Question answering



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



credit: ifunny.com

Text Classification



BIDNESS ETC

The page at <https://mail.google.com/> says:

Did you mean to attach files?
You wrote "is attached" in your message, but there are no files attached. Send anyway?

OK Cancel

1–21 of 21 < > ⚙️

Primary Social 1 new Promotions 2 new Updates 1 new

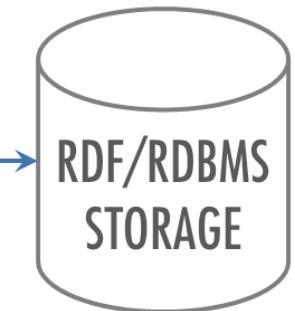
James, me (2) Hiking trip on Saturday - Yay - so glad you can join. We should leave from I 3:14 pm

Hannah Cho Thank you - Keri - so good that you and Steve were able to come over. Thank you : 3:05 pm

Jay Birdsong School Upcoming school conference dates Hello everyone. A few people have www.wired.com



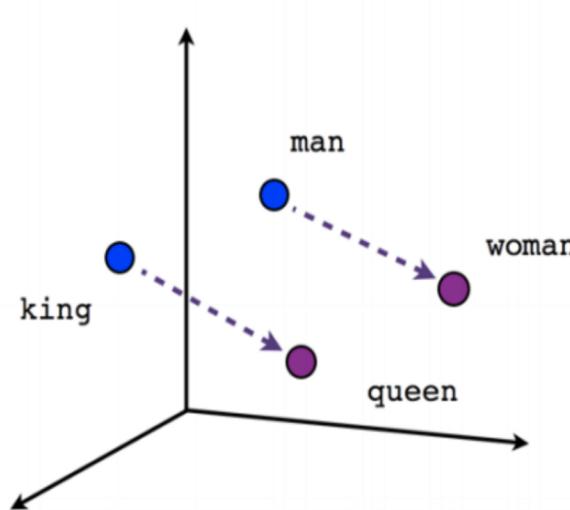
Classic NLP Pipeline
Includes a set of
Components for
Understanding Text



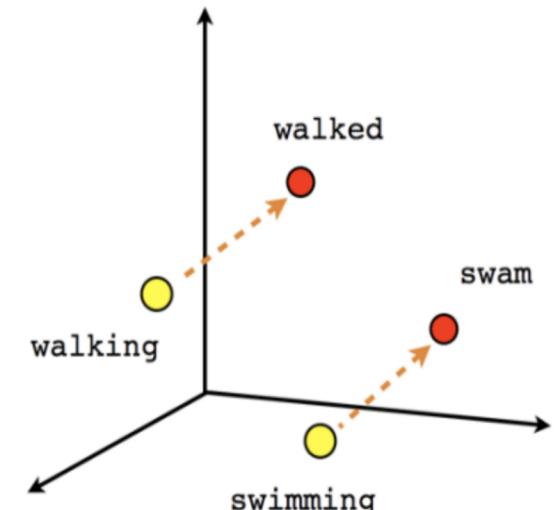
Recent deep learning advances **on** natural language

- 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

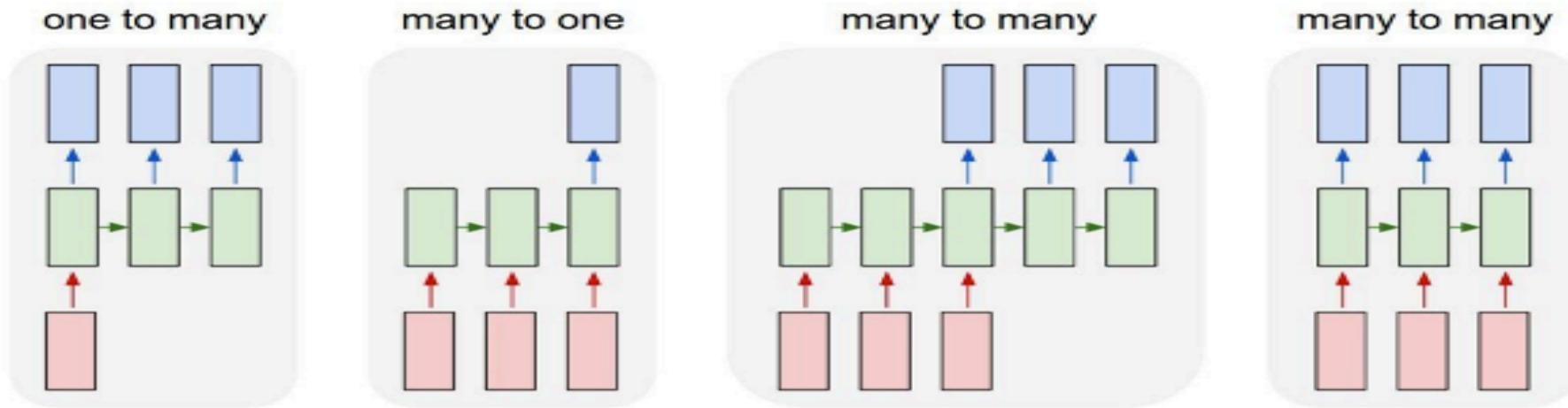


Male-Female



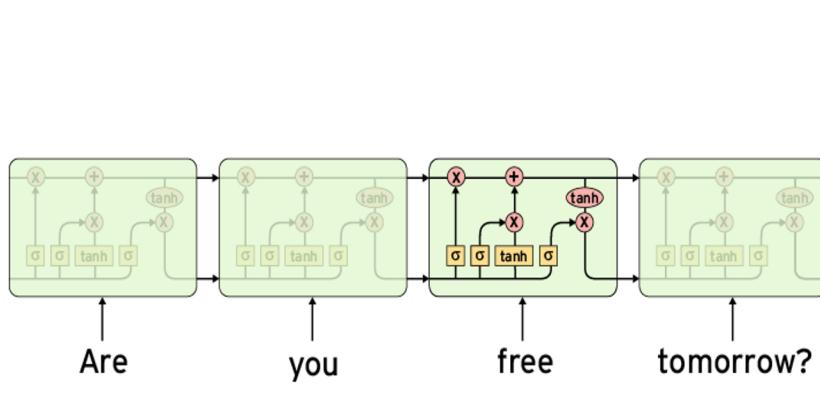
Verb tense

Recurrent Neural Networks (RNNs) can handle



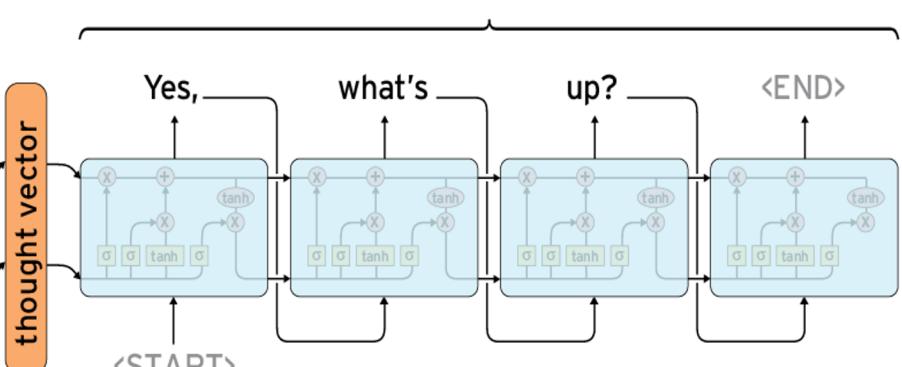
e.g. **Machine Translation**
seq of words → seq of words

ENCODER



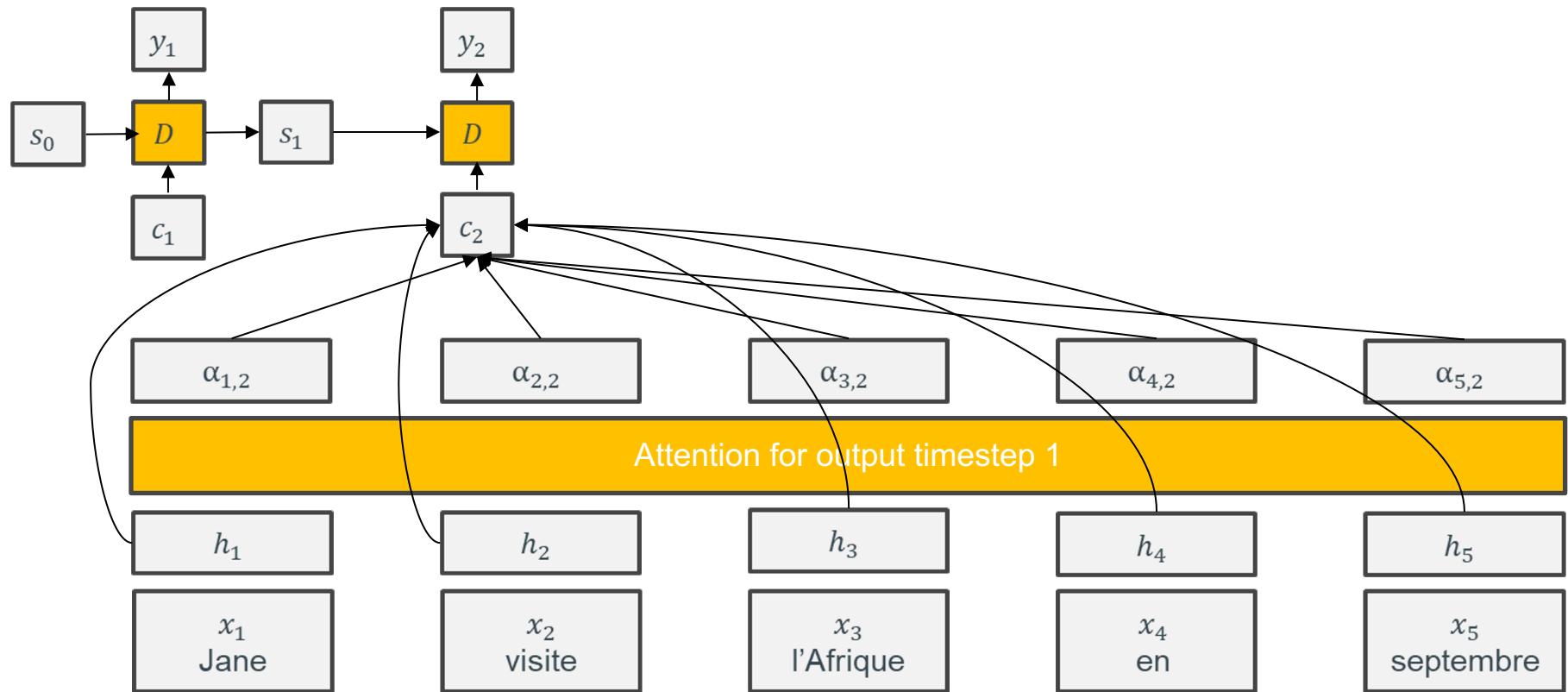
Incoming Email

Reply



DECODER

The attention module gives us a weight for each input.



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

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

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.

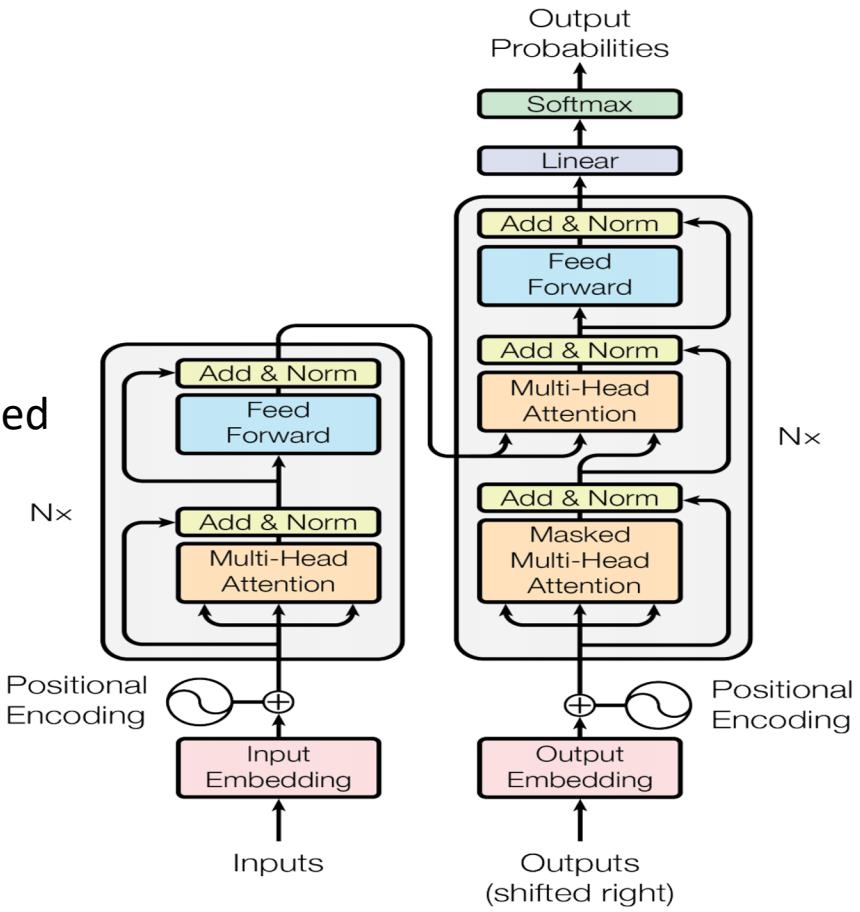


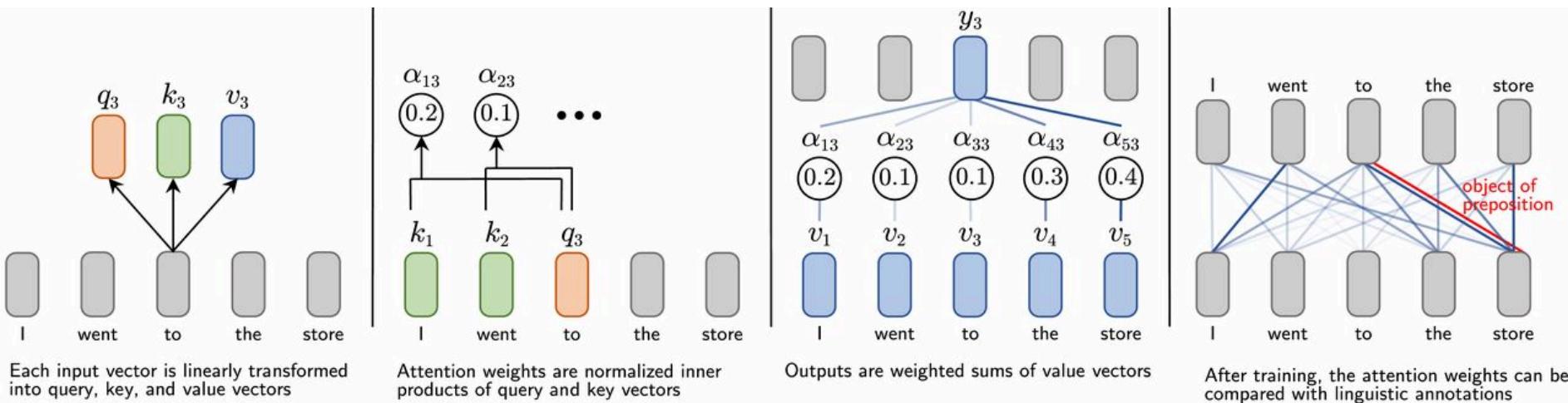
Figure 1: The Transformer - model architecture.



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

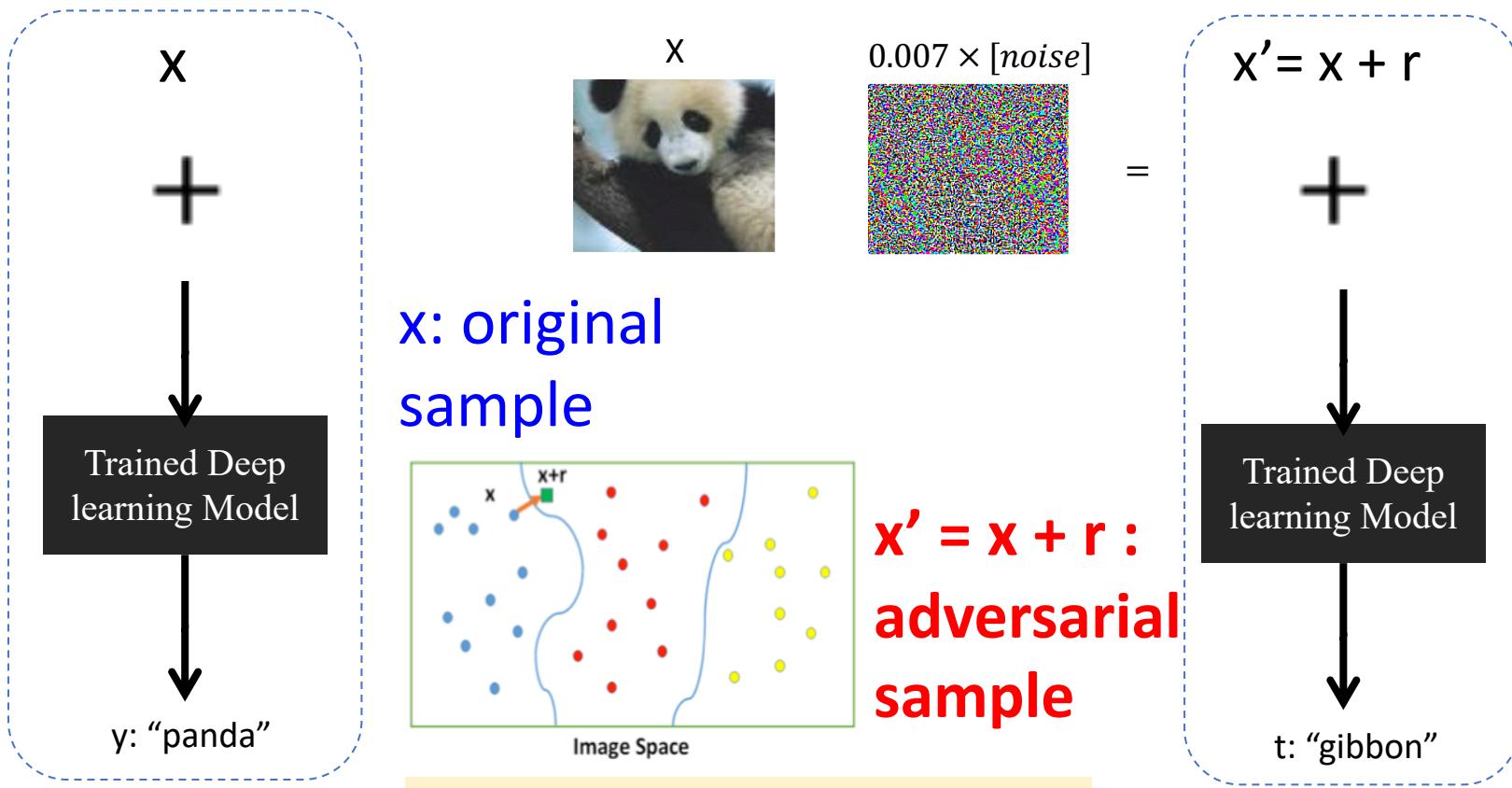


Notable pre-trained NLP models

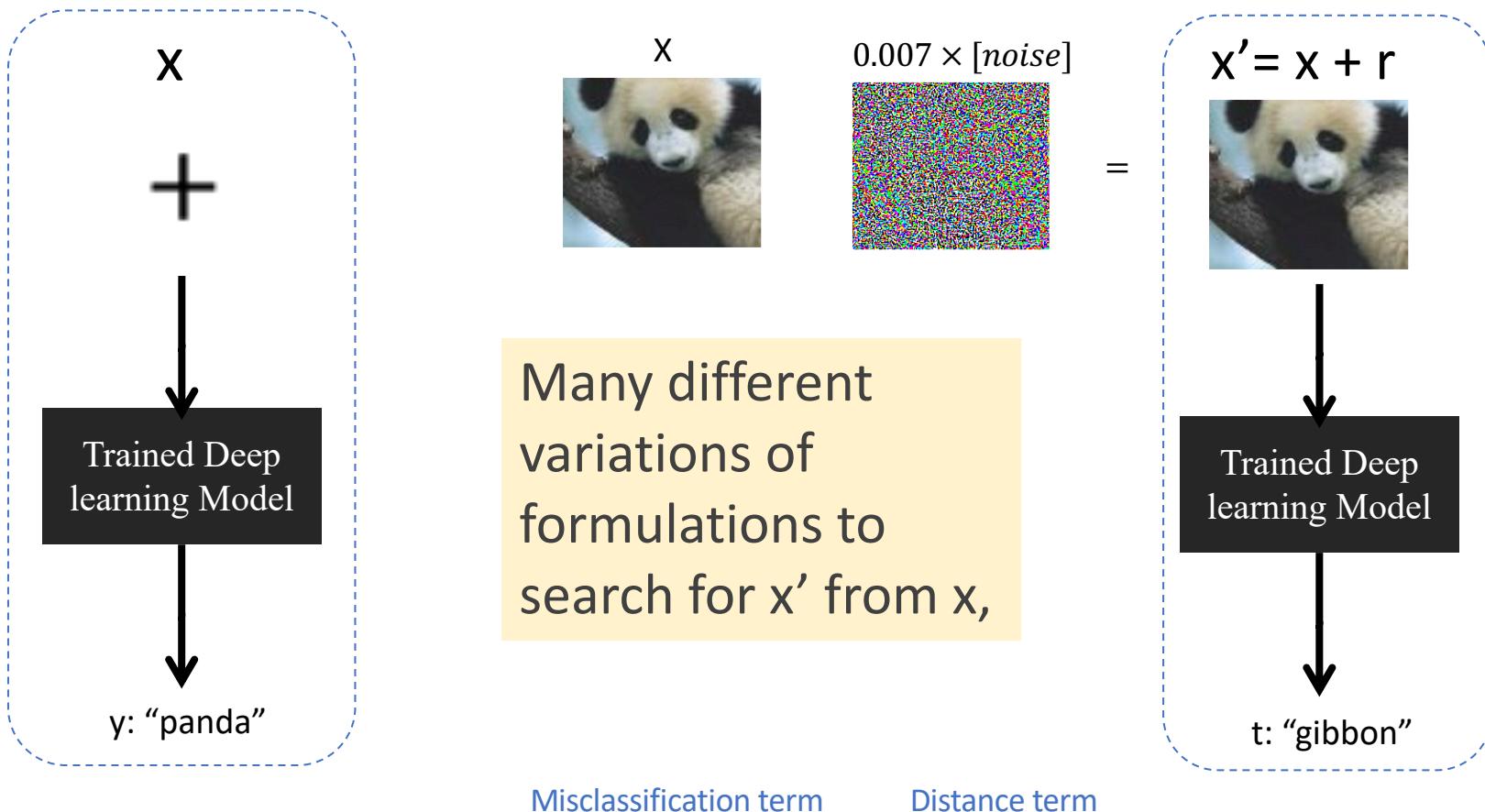


Background: Adversarial Examples

Background: Adversarial Examples



Background: Adversarial Examples



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

Misclassification term

Distance term

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

Deep Learning Classifiers are Easily Fooled

Melanoma Diagnosis with Computer Vision



Healthcare

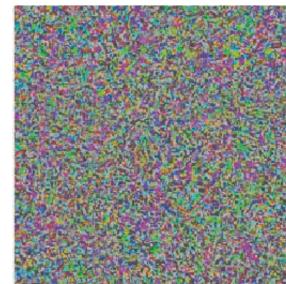
Original Image



Benign

Perturbation

+ 0.04 ×



Adversarial Example

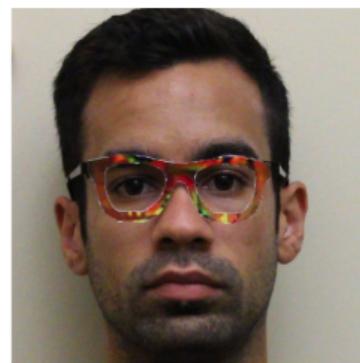
=



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

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

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

Toxicity Identification



Sentiment Classification

"I love this movie.
I've seen it many times
and it's still awesome."



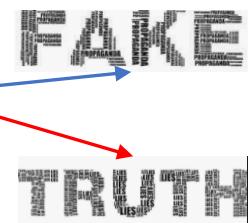
"This movie is bad.
I don't like it at all.
It's terrible."

NLP Computer System needs Trustworthiness and Robustness

Authorship Detection



Rowling?



Fake News Detection



Spam Detection



Electronic Medical Records

Misclassification term

Distance term

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

What are adversarial examples in NLP?

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

Input, x :

“True Grit” was the best **movie**
I’ve seen **since** I was a small boy.

Prediction: **Positive ✓**

Perturbation, x_{adv} :

“True Grit” was the best **moive**
I’ve seen **snice** I was a small boy.

Prediction: **Negative X**

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

24

Misclassification term

Distance term

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

What are adversarial examples in NLP?

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

Input, x :

“True Grit” was the best movie
I’ve seen since I was a **small boy**.

Prediction: **Positive ✓**

Perturbation, x_{adv} :

“True Grit” was the best movie
I’ve seen since I was a **wee lad**.

Prediction: **Negative X**



AE NLP
literature
is messy
(chaotic)

1. Many generated 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, x :

“True Grit” was the best movie I’ve seen since I was a **small boy**.

Prediction: **Positive ✓**

Perturbation, x_{adv} :

“True Grit” was the best movie I’ve seen since I was a **wee lad**.

Prediction: **Negative X**

Bad examples of adversarial perturbations in NLP

Perturbation, x_{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 **minuscule youngster**.

Constraints to ensure our transformation only produces “valid” examples?

- **Idea 1:** 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)
- **Idea 2:** Use a grammar checker to sure that we didn't introduce any grammatical errors in \mathbf{x}_{adv} .

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

$$C_1(T(x)) \wedge C_2(T(x)) \wedge \dots \wedge C_m(T(x))$$

30

Standardize Constraints Enables Better/Truthful Comparison

Constraints Search Method	TFADJUSTED		TEXTFOOLER	
	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

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

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

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

The TextAttack Framework

NLP attacks can be constructed from four components:

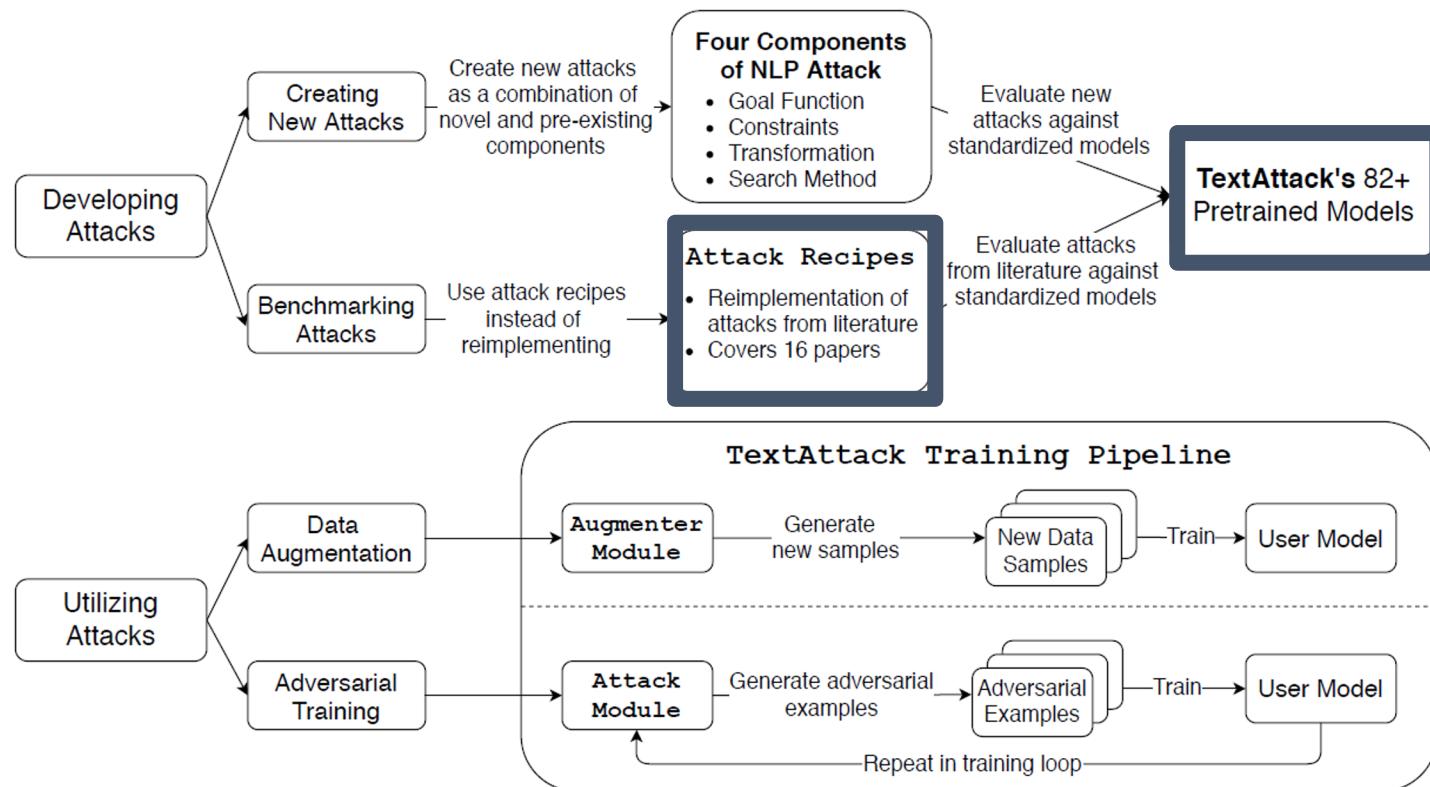
- 1. transformation** ([textattack.transformations.Transformation](#))
- 2. constraint(s)** ([list\(textattack.constraints.Constraint\)](#))
- 3. goal function** ([textattack.goal_functions.GoalFunction](#))
- 4. search method** ([textattack.search_methods.SearchMethod](#))

Goal Function term

Constraints' term

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

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 Szolovits¹

¹Computer Science & Artificial Intelligence Laboratory, MIT

²University of Hong Kong

³A*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

Classification Task: Is this a *positive* or *negative* review?

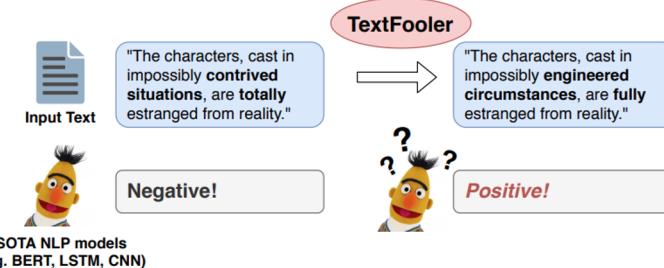


Figure 1: Our model TextFooler slightly change the input text but completely altered the prediction result.

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

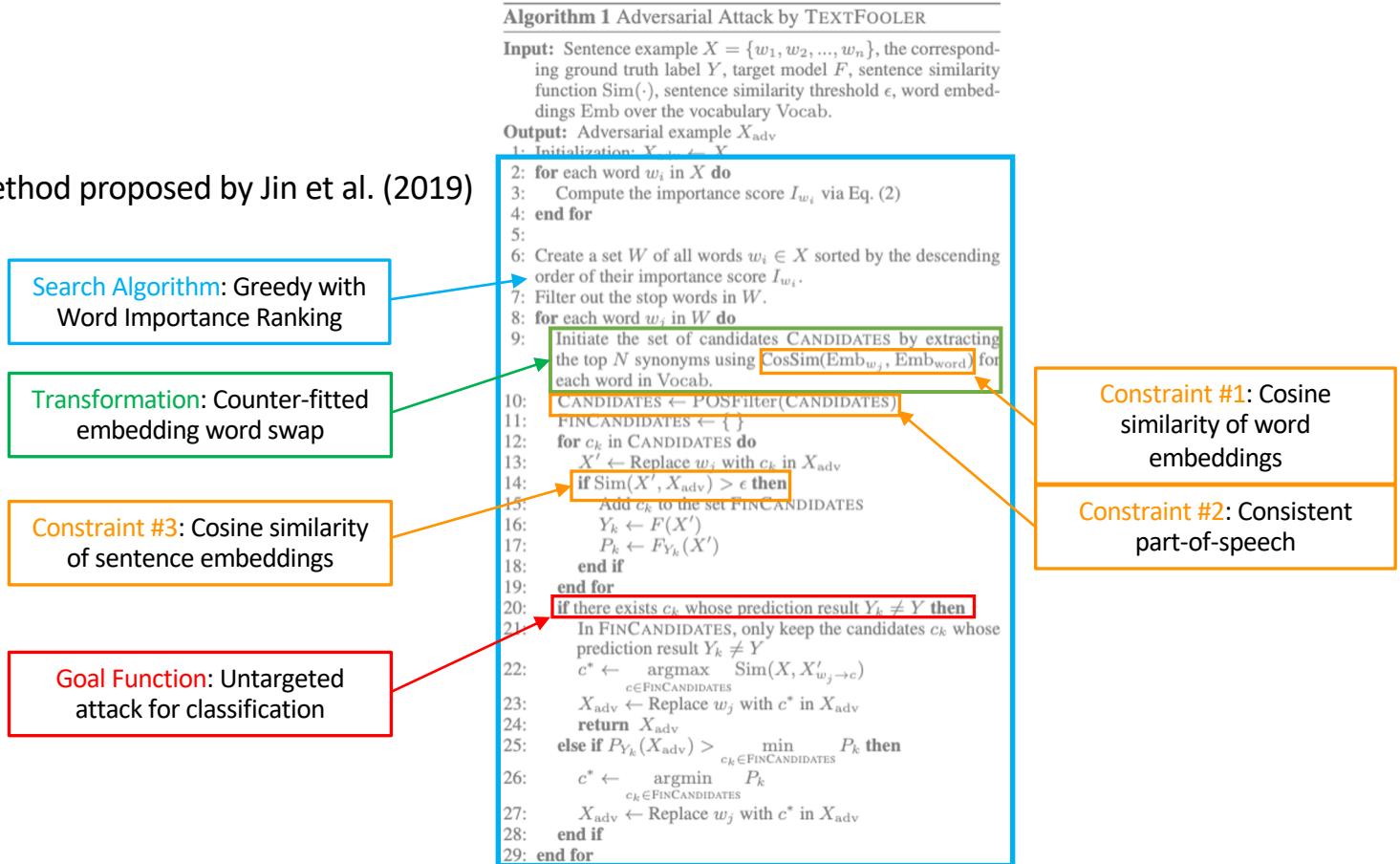
Input: Sentence example $X = \{w_1, w_2, \dots, w_n\}$, the corresponding ground truth label Y , target model F , sentence similarity function $\text{Sim}(\cdot)$, sentence similarity threshold ϵ , word embeddings Emb over the vocabulary Vocab .

Output: Adversarial example X_{adv}

- 1: Initialization: $X_{\text{adv}} \leftarrow X$
- 2: **for** each word w_i in X **do**
- 3: Compute the importance score I_{w_i} via Eq. (2)
- 4: **end for**
- 5:
- 6: Create a set W of all words $w_i \in X$ sorted by the descending order of their importance score I_{w_i} .
- 7: Filter out the stop words in W .
- 8: **for** each word w_j in W **do**
- 9: Initiate the set of candidates CANDIDATES by extracting the top N synonyms using $\text{CosSim}(\text{Emb}_{w_j}, \text{Emb}_{\text{word}})$ for each word in Vocab .
- 10: $\text{CANDIDATES} \leftarrow \text{POSFilter}(\text{CANDIDATES})$
- 11: $\text{FINCANDIDATES} \leftarrow \{\}$
- 12: **for** c_k in CANDIDATES **do**
- 13: $X' \leftarrow \text{Replace } w_j \text{ with } c_k \text{ in } X_{\text{adv}}$
- 14: **if** $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ **then**
- 15: Add c_k to the set FINCANDIDATES
- 16: $Y_k \leftarrow F(X')$
- 17: $P_k \leftarrow F_{Y_k}(X')$
- 18: **end if**
- 19: **end for**
- 20: **if** there exists c_k whose prediction result $Y_k \neq Y$ **then**
- 21: In FINCANDIDATES , only keep the candidates c_k whose prediction result $Y_k \neq Y$
- 22: $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \text{Sim}(X, X'_{w_j \rightarrow c})$
- 23: $X_{\text{adv}} \leftarrow \text{Replace } w_j \text{ with } c^* \text{ in } X_{\text{adv}}$
- 24: **return** X_{adv}
- 25: **else if** $P_{Y_k}(X_{\text{adv}}) > \min_{c_k \in \text{FINCANDIDATES}} P_k$ **then**
- 26: $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
- 27: $X_{\text{adv}} \leftarrow \text{Replace } w_j \text{ with } c^* \text{ in } X_{\text{adv}}$
- 28: **end if**
- 29: **end for**
- 30: **return** None

Four Components in Action

TextFooler method proposed by Jin et al. (2019)



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	<ul style="list-style-type: none">WordsPerturbedPercentage(max_perc=20)WordEmbeddingDistance(max_mse=0.5)GoogleLanguageModel(n_per_index=4)	<ul style="list-style-type: none">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>

QData / TextAttack

 Unwatch 28  Star 1.4k  Fork 158

 Code

 Issues 35

 Pull requests 9

 Actions

 Projects 6

 Wiki

 Security

 Insights

 Settings

 master

 22 branches

 9 tags

 Go to file

 Add file

 Code



qiyanjun Update README.md ...

 ae68c81 5 days ago  1,989 commits

 .github

Update run-pytest.yml

2 months ago

 docs

Fix errors in Example_5_Explain_BERT

6 days ago

 examples

isort format of attack_camerbert

6 months ago

 tests

Revert "add --split to specify train/test/dev dataset"

12 days ago

 textattack

Revert "add --split to specify train/test/dev dataset"

12 days ago

 .gitignore

delete vscode setting

7 months ago

 .readthedocs.yml

fix readthedocs module load

5 months ago

 CONTRIBUTING.md

Clarify CONTRIBUTING.md

8 months ago

 LICENSE

Initial commit

2 years ago

 Makefile

autobuild cli changed

13 days ago

 README.md

Update README.md

5 days ago

 README_ZH.md

correct the EMNLP BlackBoxNLP mentions.

4 months ago

 pytest.ini

update travis for jenkins

10 months ago

 requirements.txt

locally test all passed...

9 days ago

 setup.cfg

merge in master and fix syntax errors

10 months ago

 setup.py

Update setup.py

9 days ago

About

TextAttack  is a Python framework for adversarial attacks, data augmentation, and model training in NLP

 <textattack.readthedocs.io/en/latest/>

nlp security machine-learning

natural-language-processing

data-augmentation

adversarial-machine-learning

adversarial-examples

adversarial-attacks

 Readme

 MIT License

Releases 9

 v0.2.15: CLARE Attack, Cu...

Latest

on Dec 26, 2020 + 8 releases

Packages

No packages published

AE NLP
literature
is messy
(chaotic)

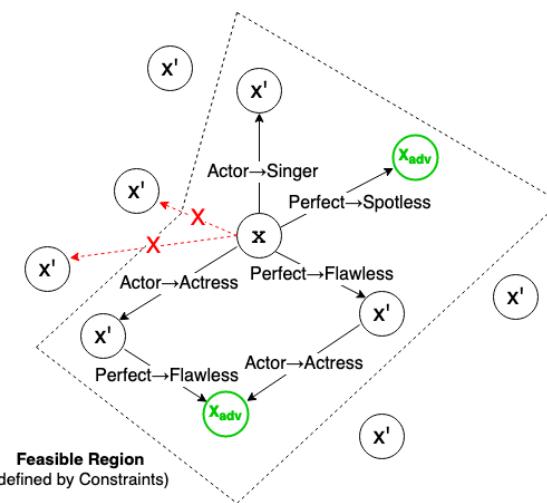
3. No clear benchmarking
insights

Search Algorithm

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

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](#)

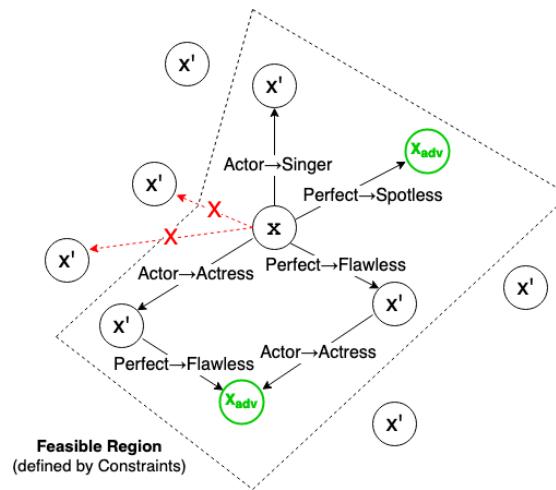
Search Space

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

Search space defined by transformation and constraints

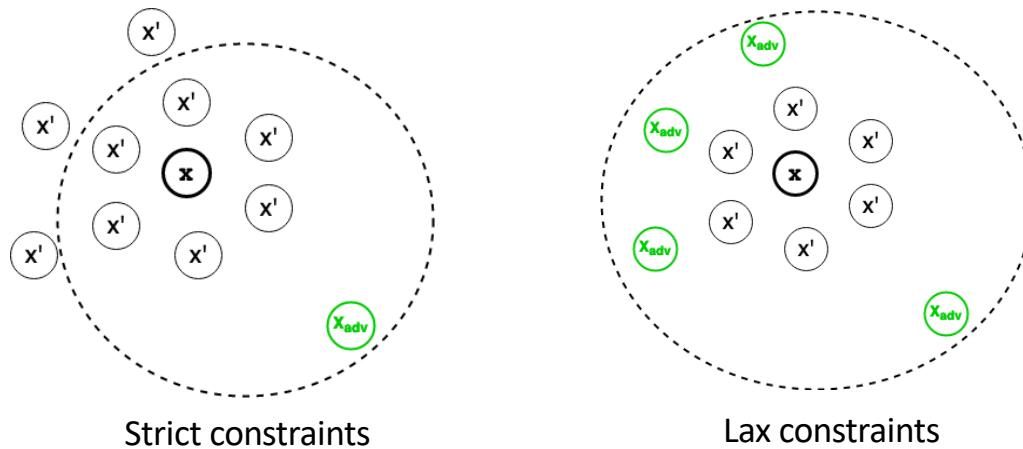
Let $T(x)$ be our transformation and $C_i(x)$ be a constraint,

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



Search Space

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



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
BERT	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
BERT	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

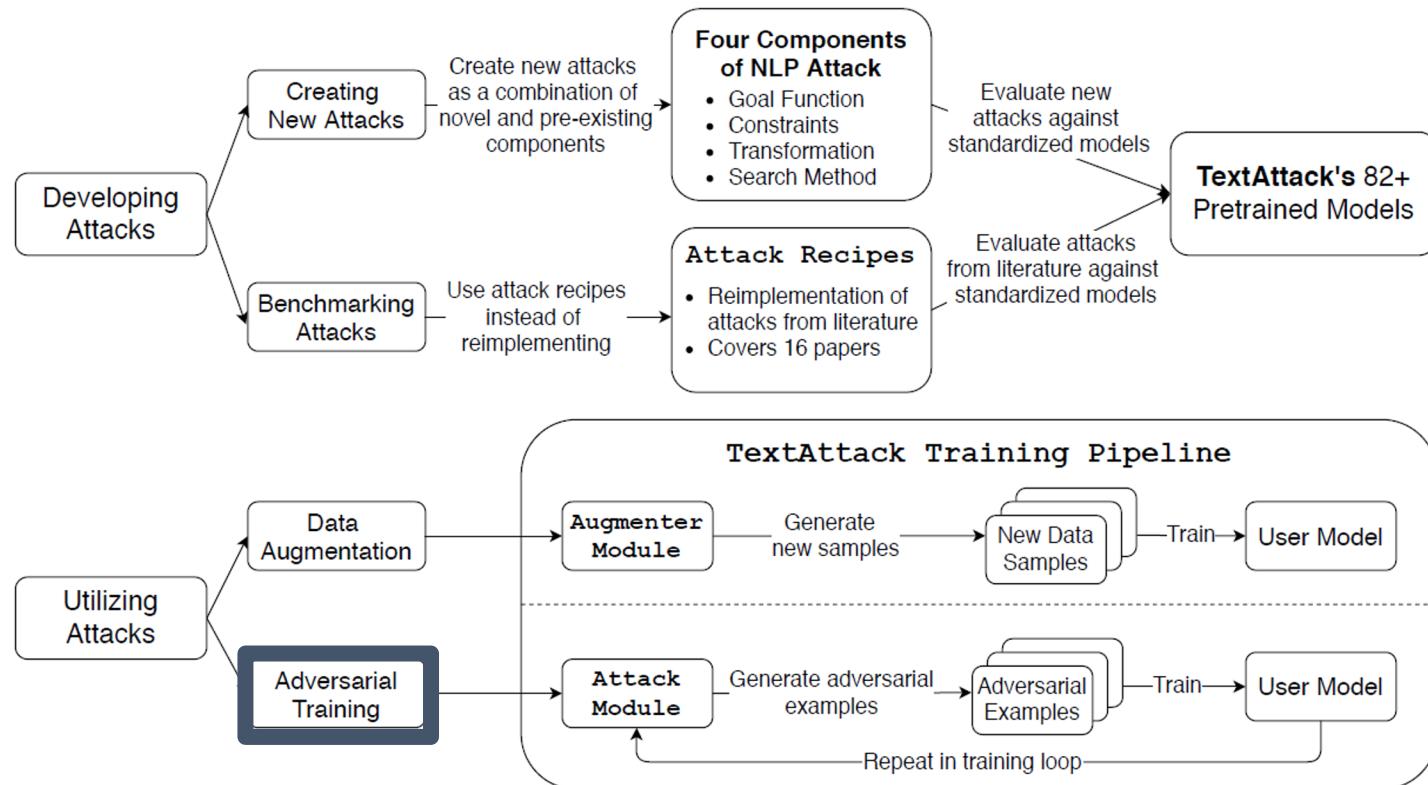
Benchmarking Insights

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

$$\text{Madry et al. (2017):} \quad \arg \min_{\theta} \mathbb{E}_{(x,y) \in \hat{p}_{\text{data}}} \left(\max_{\delta \in S} L(\theta, x + \delta, y) \right) \quad (1)$$

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

Kannan et al. (2018):

IMDB-BERT Results

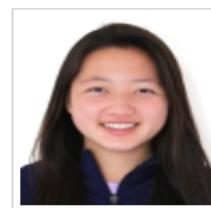
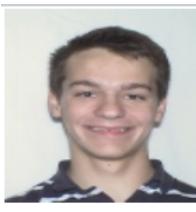
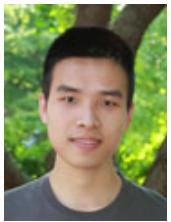
	IMDB Test Acc	Yelp Test Acc	Counterfactual Acc
No adv. training	93.97	92.86	92.84
SSMBA	93.94	92.52	92.48
Backtranslation	93.97	92.62	92.58
Textfooler-Mod	94.49	93.29	93.23
BAE-Mod	93.05	91.61	91.35

TextAttack Rescues Messy AE NLP literature

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

Acknowledgements

My Students on this project:



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

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

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

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



4/14/21

Yanjun Qi / UVA CS



58

<http://trustworthymachinelearning.org>

1. To Fool / Evade Learned Models

2. To Detect fooling/ Evasion

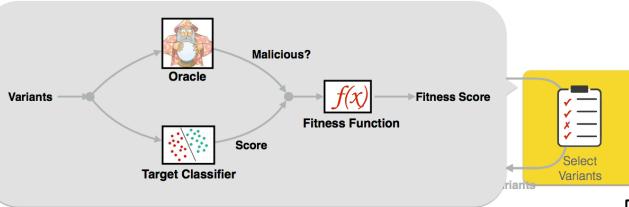
3. To Defend Against Evasion

4. To Visualize and Benchmarking

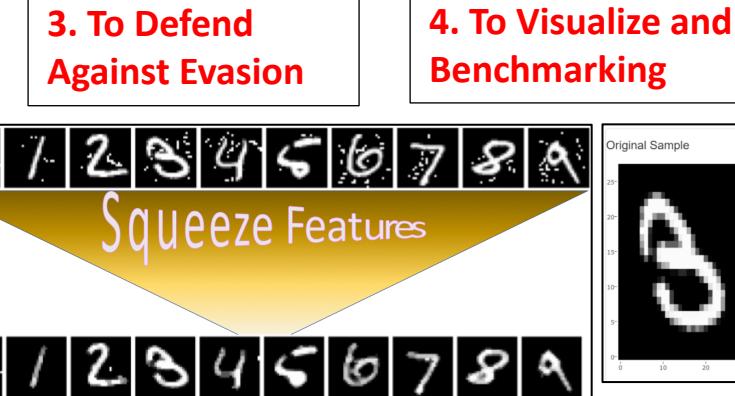
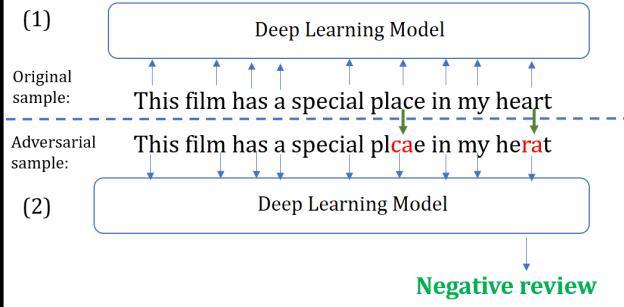
5. To Understand Theoretically

Automated Evasion Approach

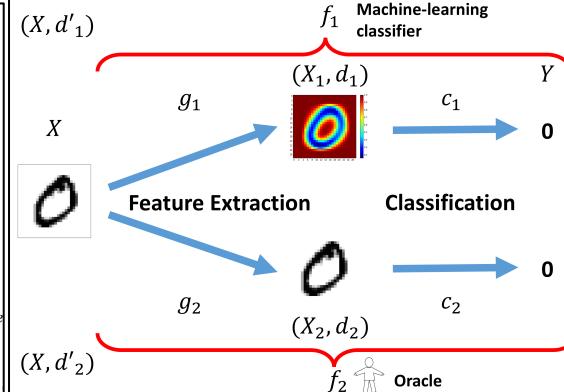
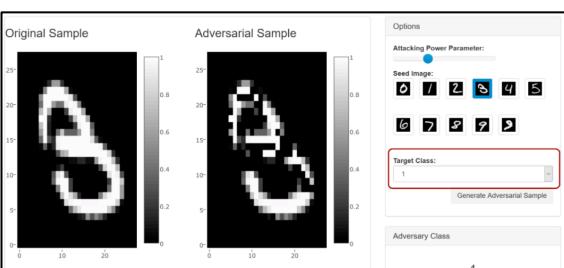
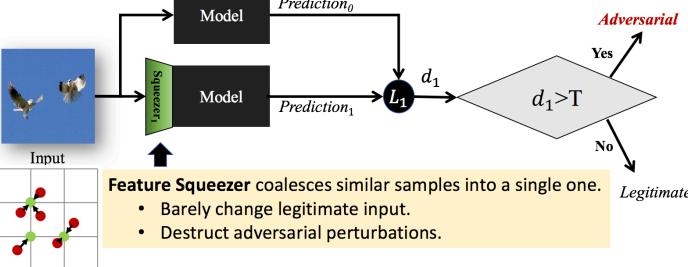
Based on Genetic Programming

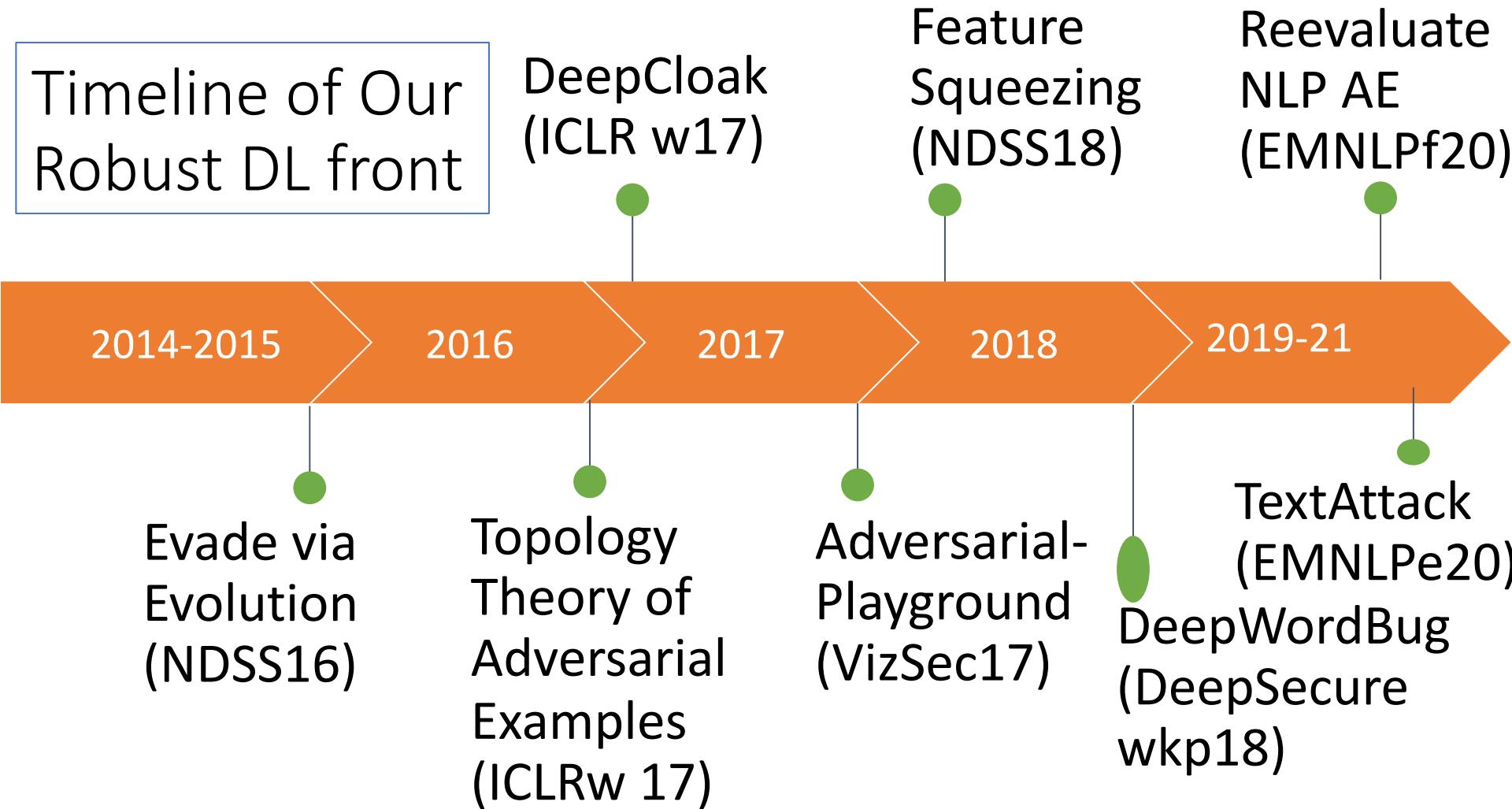


Positive review



Detection Framework







Thank you