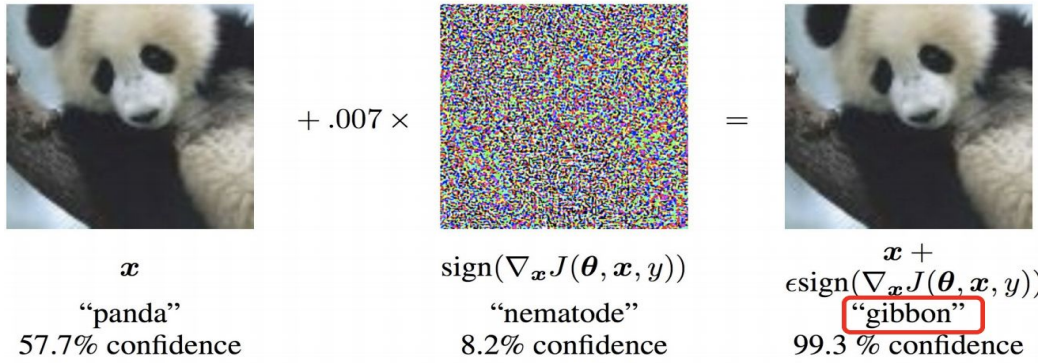
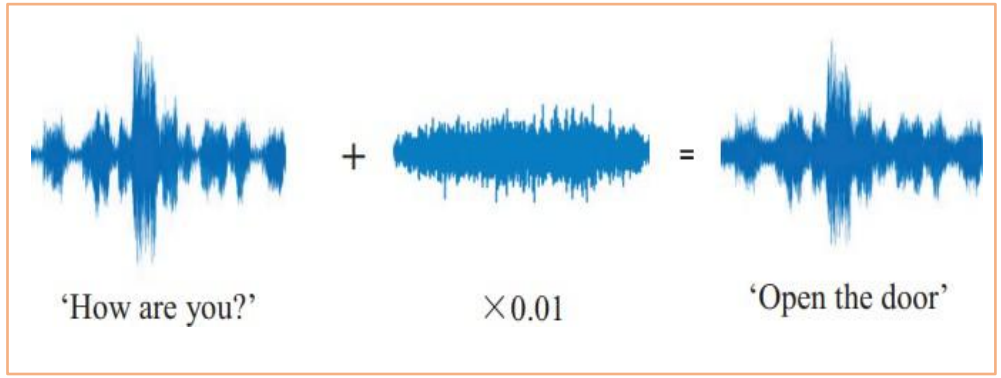
Generating Natural Language Attacks in a Hard Label Black Box Setting

Authors: Rishabh Maheshwary, Saket Maheshwary and Vikram Pudi
International Institute of Information Technology
Hyderabad

Adversarial Examples in AI

Adversarial examples are carefully crafted inputs to Deep Learning Models that an attacker has intentionally designed to cause the model to make a mistake.



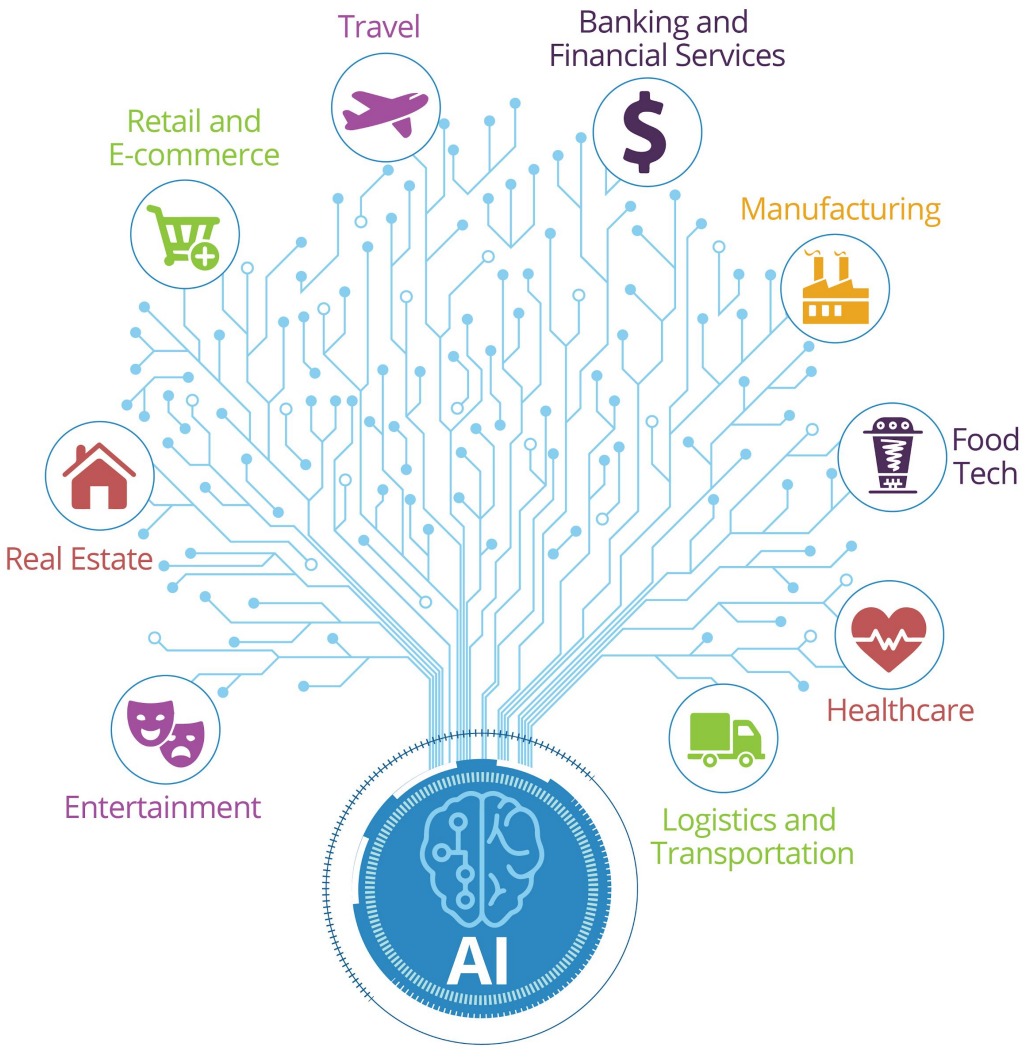


Original Text Prediction = **Negative**. (Confidence = 78.0%)
*This movie had **terrible** acting, **terrible** plot, and **terrible** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **considered** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **kids** they didn't understand that theme.*
Adversarial Text Prediction = **Positive**. (Confidence = 59.8%)
*This movie had **horrific** acting, **horrific** plot, and **horrifying** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **regarded** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **youngsters** they didn't understand that theme.*

Such adversarial examples are imperceptible to humans but they deceive Deep Neural Networks.

Why Study Adversarial Examples ?

- Deep Neural Networks are used in a variety of real world applications.
- The existence of these adversarial examples pose a threat to the security of deep neural networks deployed in real world.
- Hence it is important to study the effect of adversarial attacks on deep neural networks in different real world scenarios.



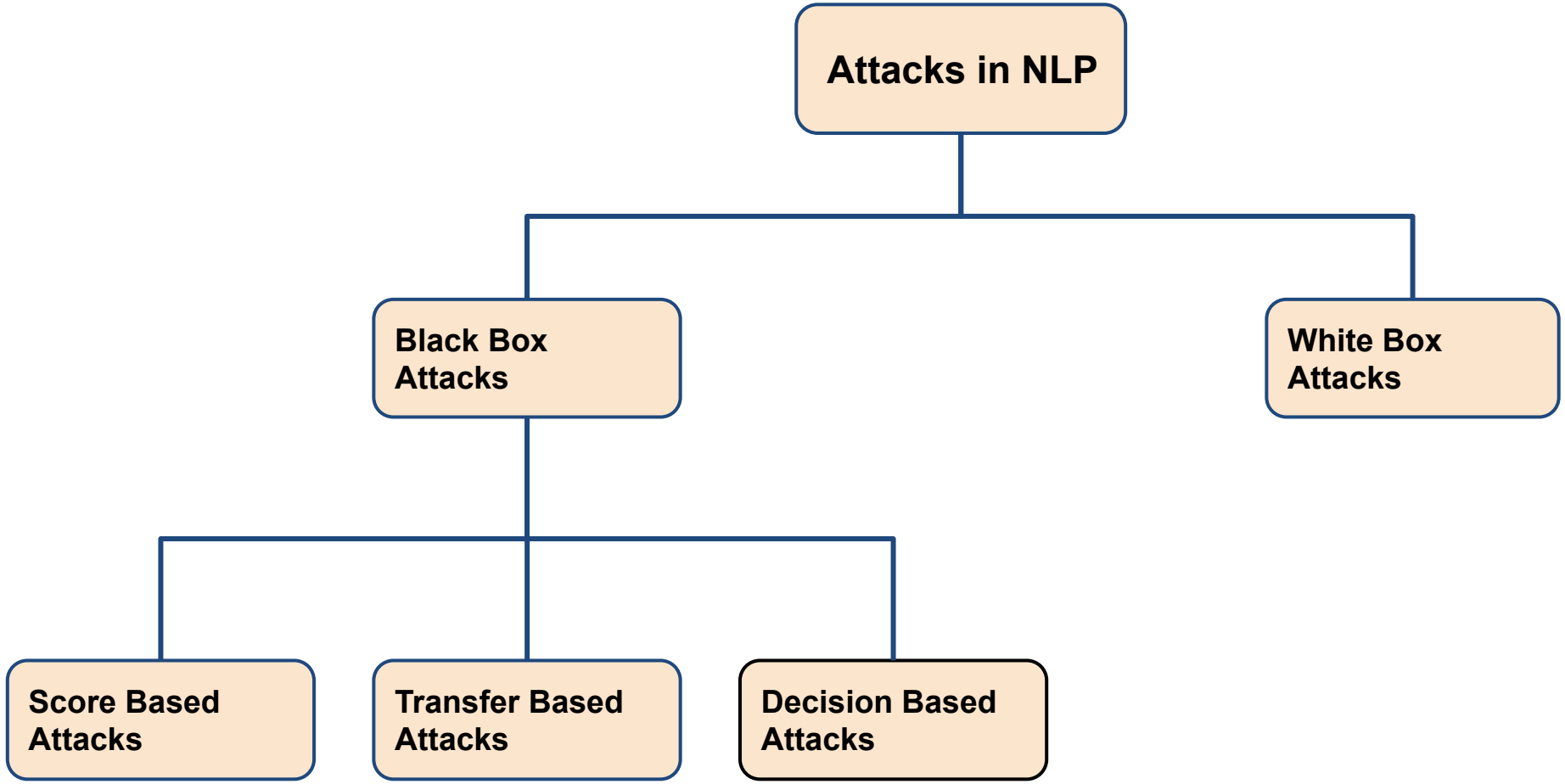
Adversarial Attacks in NLP vs CV

How generating adversarial examples in NLP is different from CV?

1. **Discrete Input:** Replacing a single word in a text can completely alter its semantics.
2. **Perceivable:** Small perturbations in text are usually clearly perceptible.
3. **Grammatical correctness and fluency:** Text should be grammatically correct and fluent.

Therefore generating adversarial examples in NLP is challenging compared to other domains.

Classification of Attacks in NLP



White Box Attacks in NLP

- White Box attacks require access to complete knowledge about the target model.
- Such attack methods rely on gradient information of the loss w.r.t input to find generate an attack.

Existing attack methods

- HotFlip: White-Box Adversarial Examples for Text Classification. ACL 2018.
- TEXTBUGGER: Generating Adversarial Text Against Real-world Applications. NDSS 2019.
- Deep Text Classification Can be Fooled. IJCAI 2018.
- Universal Adversarial Triggers for Attacking and Analyzing NLP. EMNLP-IJCNLP 2019.

Drawbacks:

- Requires access to complete information of the target model which is not available in real world applications.
- Such methods are slow and computationally expensive.

Score based Attacks in NLP

- Score based attacks require access to target model's confidence scores or class probabilities.
- Such methods generate an attack by first finding important words and replace them with similar words.

Existing Attacks:

- Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers
- Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. ACL 2019.
- Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. AAAI 2020.
- Generating Natural Language Adversarial Examples. EMNLP 2018
- Word-level Textual Adversarial Attacking as Combinatorial Optimization. ACL 2020.

Drawbacks:

- Require access to confidence scores which may not be available in real world applications.
- The confidence scores can be wrong to avoid attacks.

Transfer based Attacks in NLP

- Transfer-based attacks rely on information about the training data on which the target models are trained.
- Such attacks train a substitute model to mimic the decision boundary of target classifier. Adversarial attacks are generated against this substitute model and transferred to target model.

Prior attacks

- Generating Black-Box Adversarial Examples for Text Classifiers Using a Deep Reinforced Model. ECML-PKDD 2019.
- Practical Black-Box Attacks against Machine Learning. ACM 2017.

Drawbacks

- Transfer-based attacks are expensive to apply as they require training a substitute model.
- Transfer-based attacks rely on information about the training data on which the target models are trained.
- It also relies on the assumption that adversarial examples successfully transfer between models of different architectures.

Decision based Attacks in NLP

- Decision-based attacks only depends on the top predicted label of the target classifier.
- Compared to all the above attack strategies, generating adversarial examples in this strategy is most challenging.

Prior works:

- Decision Based Adversarial Attacks: Reliable Attacks Against BlackBox Machine Learning Models. ICLR 2018. (Computer Vision)
- Query-efficient hard-label black-box attack: An optimization-based approach. ICLR 2019. (Computer Vision)

The only relevant prior NLP decision-based attack Generating natural adversarial examples (Zhao, Dua, and Singh 2017) uses Generative Adversarial Network (GANs) which are very hard to train and require access to training data.

We refer to this as hard label black box setting.

Problem Statement

Find an adversarial text that is misclassified by the target model and is semantically similar to the original input.

- $F(X) \neq F(X^*)$
  - X Original text input
  - X* Adversarial text input
  - F Target model
- X* should be Semantically Similar to X

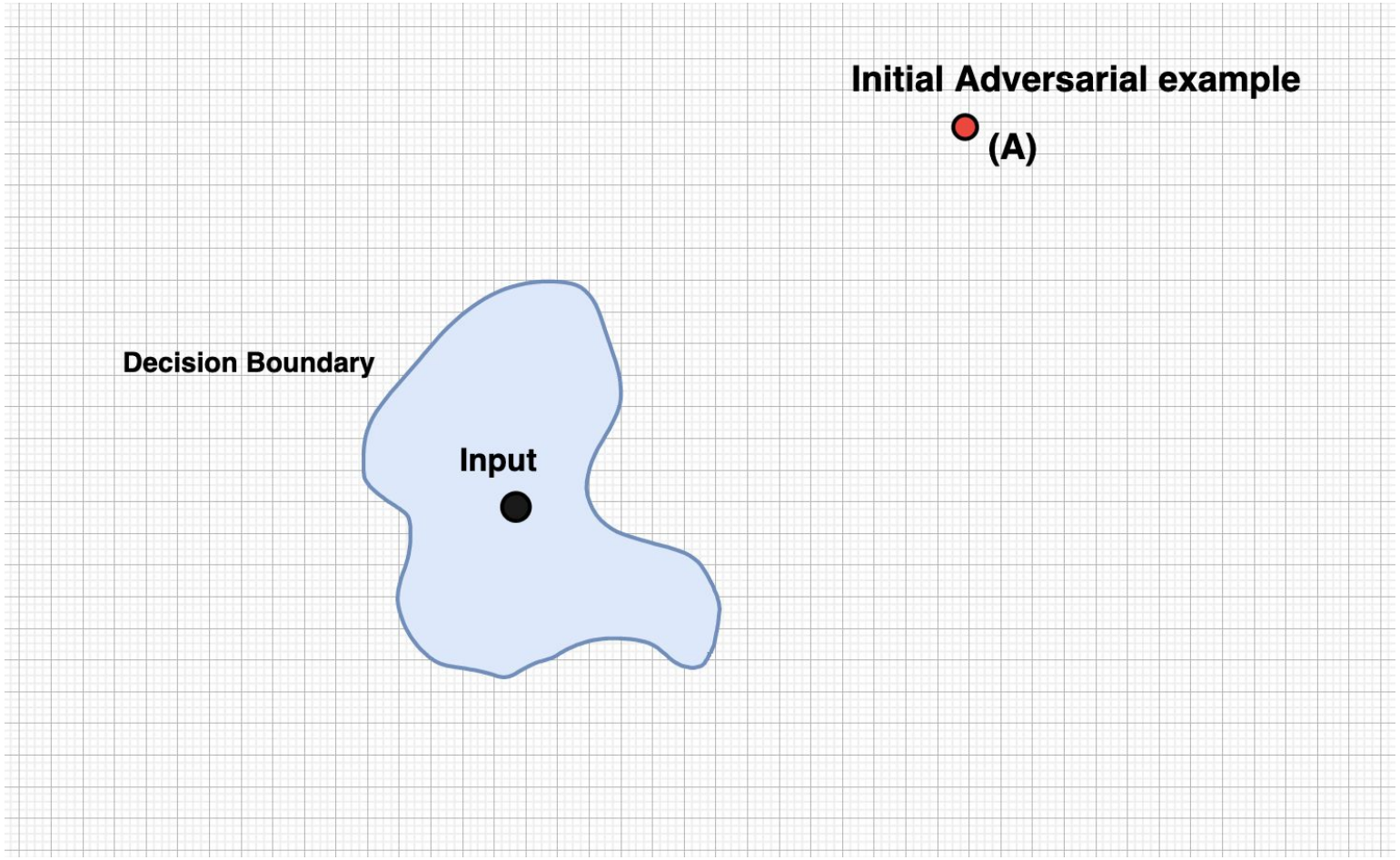
In a Hard label black-box setting where the attacker has no access to:

- **Model's gradients, parameters.**
- **The confidence scores of the target model.**
- **The training data on which the target models are trained.**

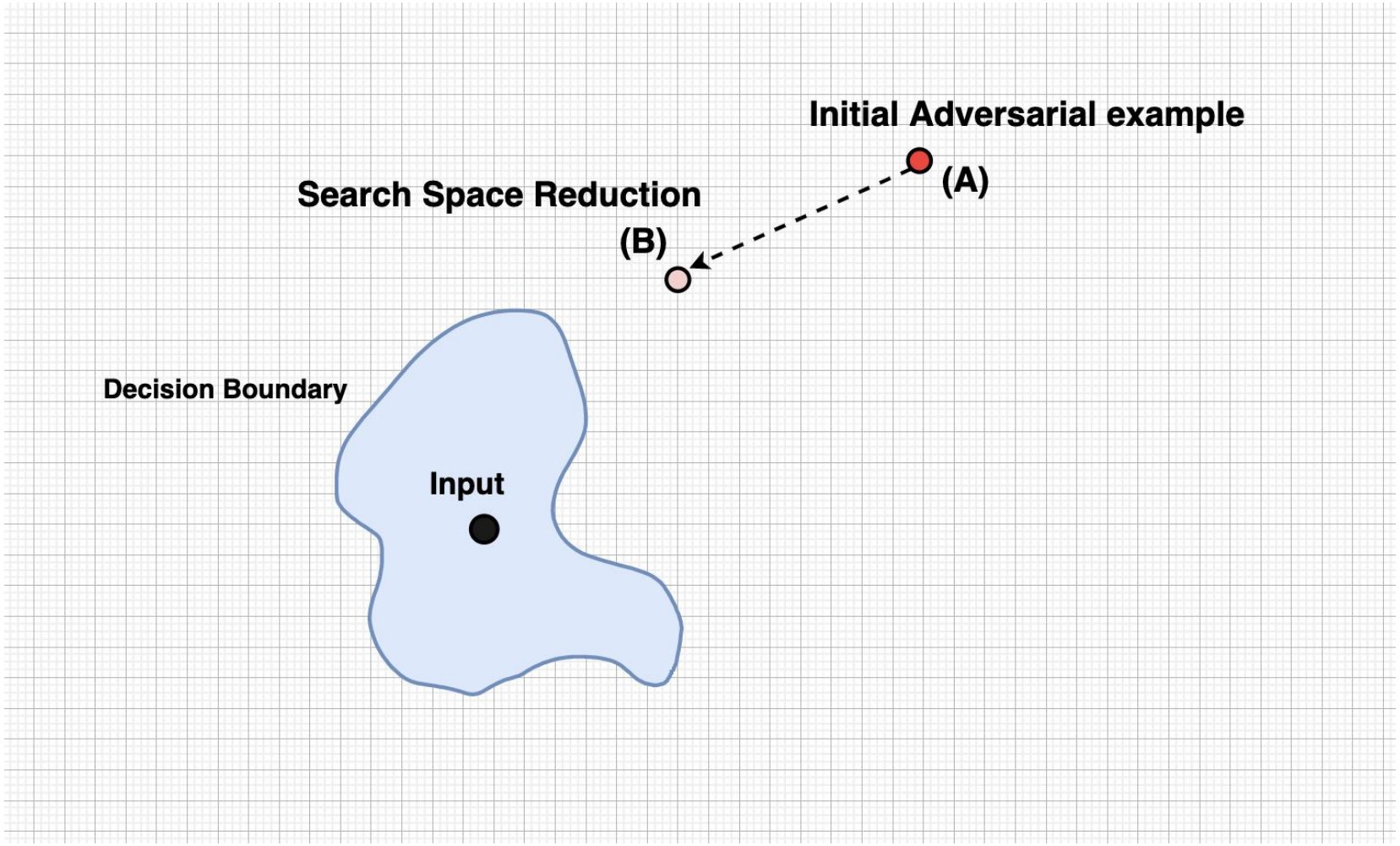
The attacker has access to:

- **Just the top label.**

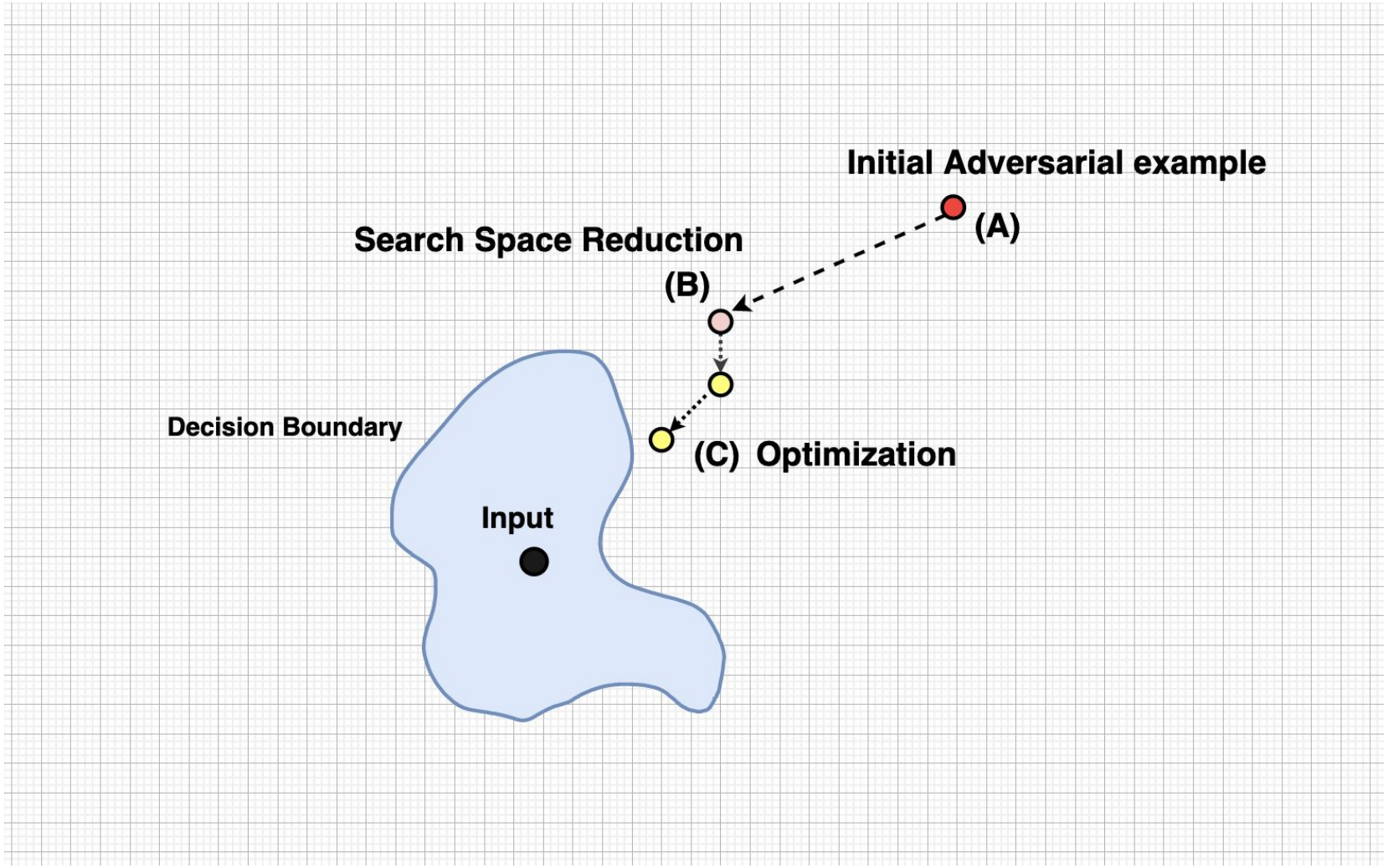
Overview



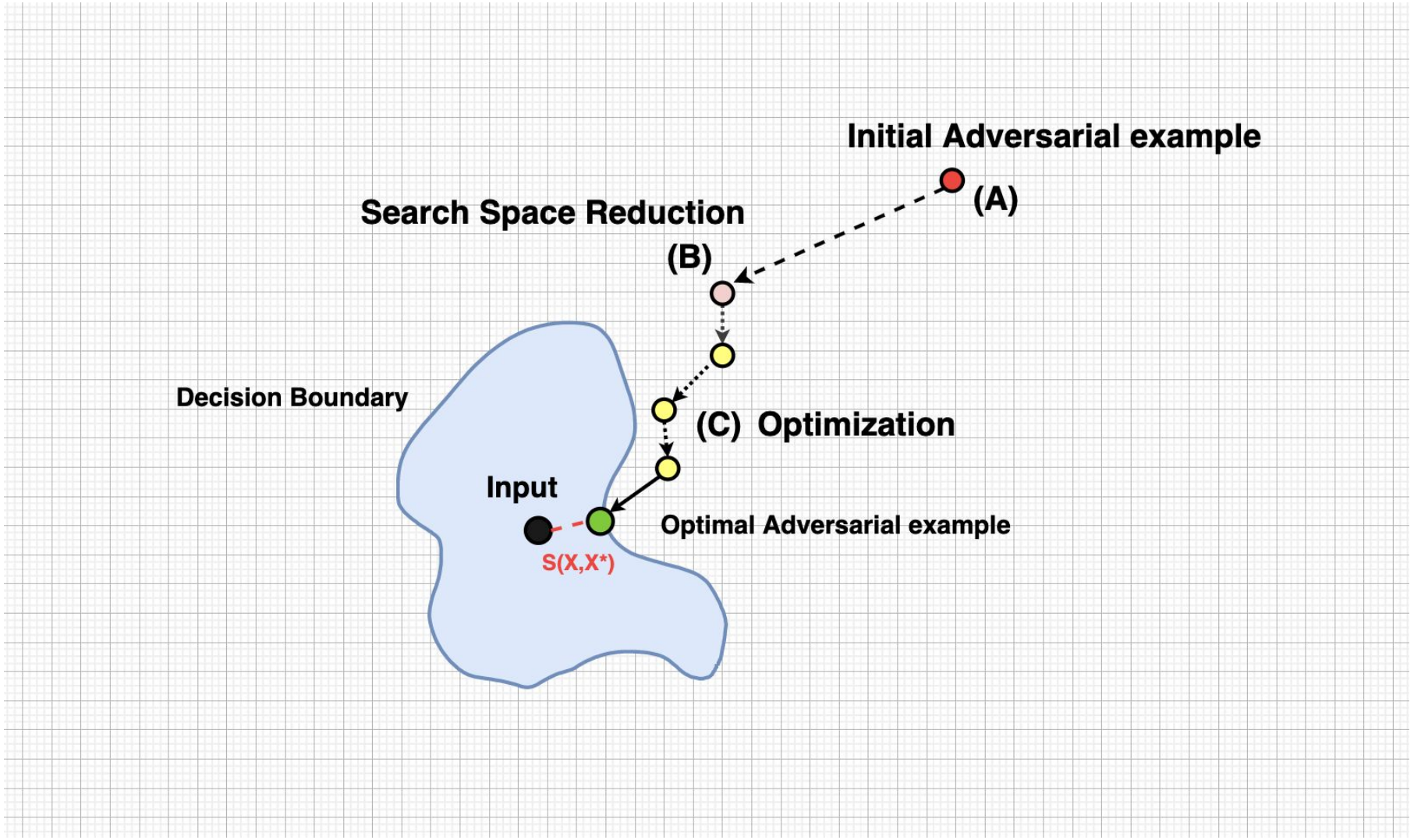
Overview


Initial Adversarial example
(A)
Search Space Reduction
(B)
Decision Boundary
Input

Overview


Initial Adversarial example
(A)
Search Space Reduction
(B)
Decision Boundary
(C) Optimization
Input

Overview


Initial Adversarial example
(A)
Search Space Reduction
(B)
Decision Boundary
(C) Optimization
Input
Optimal Adversarial example
S(X,X*)

Overview

Objective Function

$$\max_{x^*} S(X, X^*) \text{ s.t. } C(\mathbf{F}(X^*)) = 1$$

where

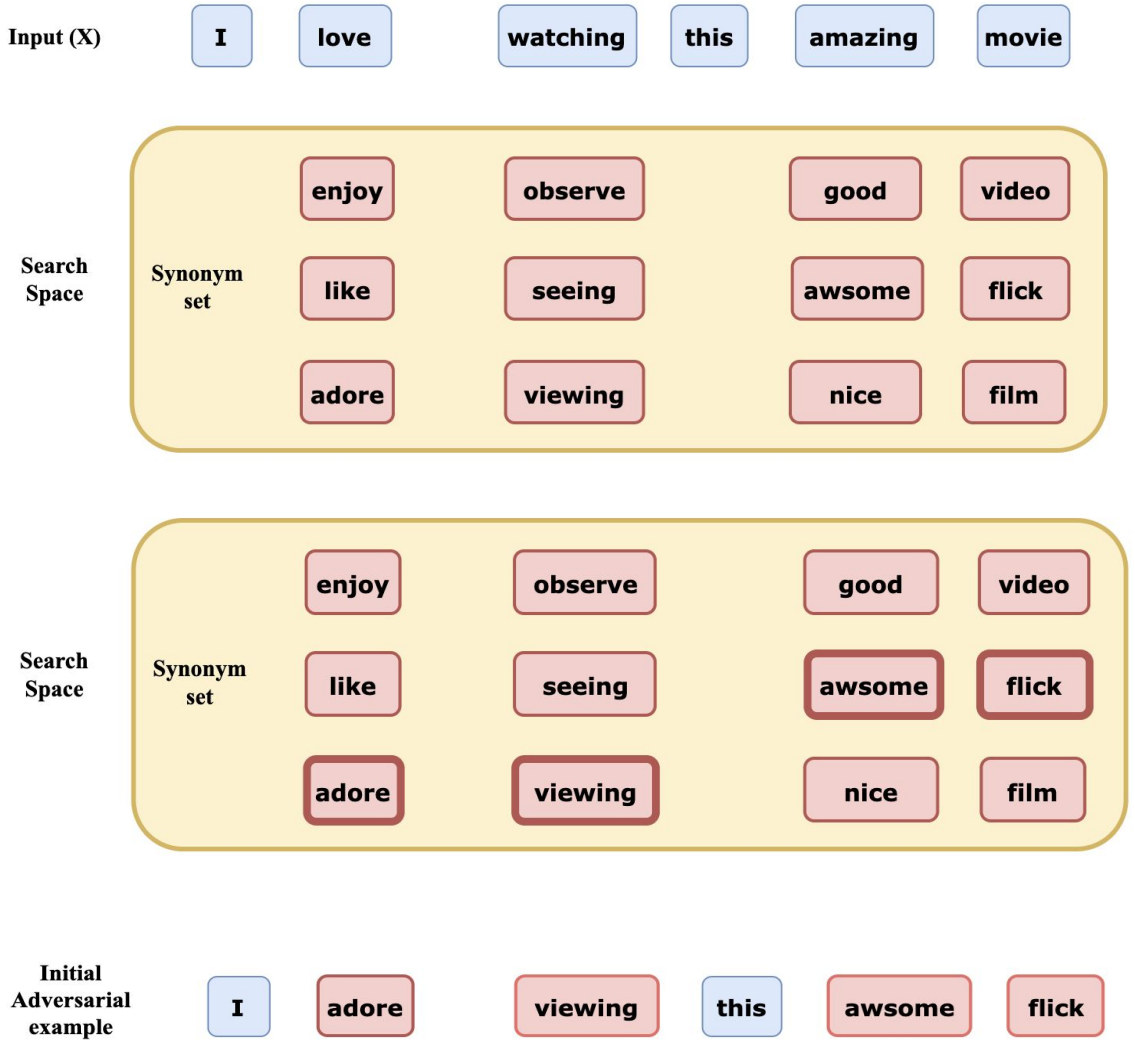
- S Semantic Similarity between two text inputs
- X Original text input
- X^* Adversarial text input
- C Adversarial Criteria which is 1 if X* is outside of target model's decision boundary otherwise 0
- F Target model

Step 1: Finding Initial Adversarial Example

This steps finds an initial adversarial example which is out of target model's decision boundary.

- ❏ It randomly selects a word to replace.
- ❏ It replace the word with a synonym selected randomly.

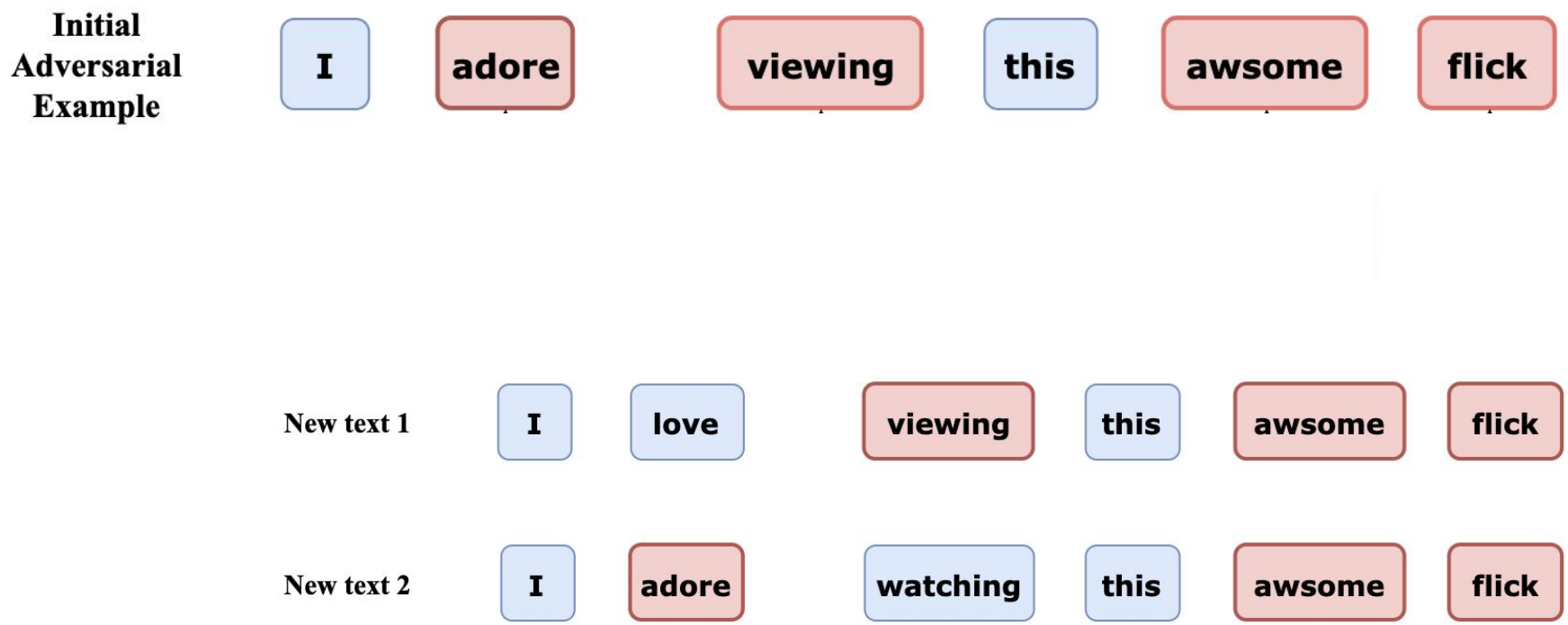
The above steps repeats till the text input moves out of target model's decision boundary or certain percentage of words have been substituted.



Step 2: Search Space Reduction

This steps reduces the number of substituted words in the initial adversarial example by replacing back the substituted words with their original counterparts.

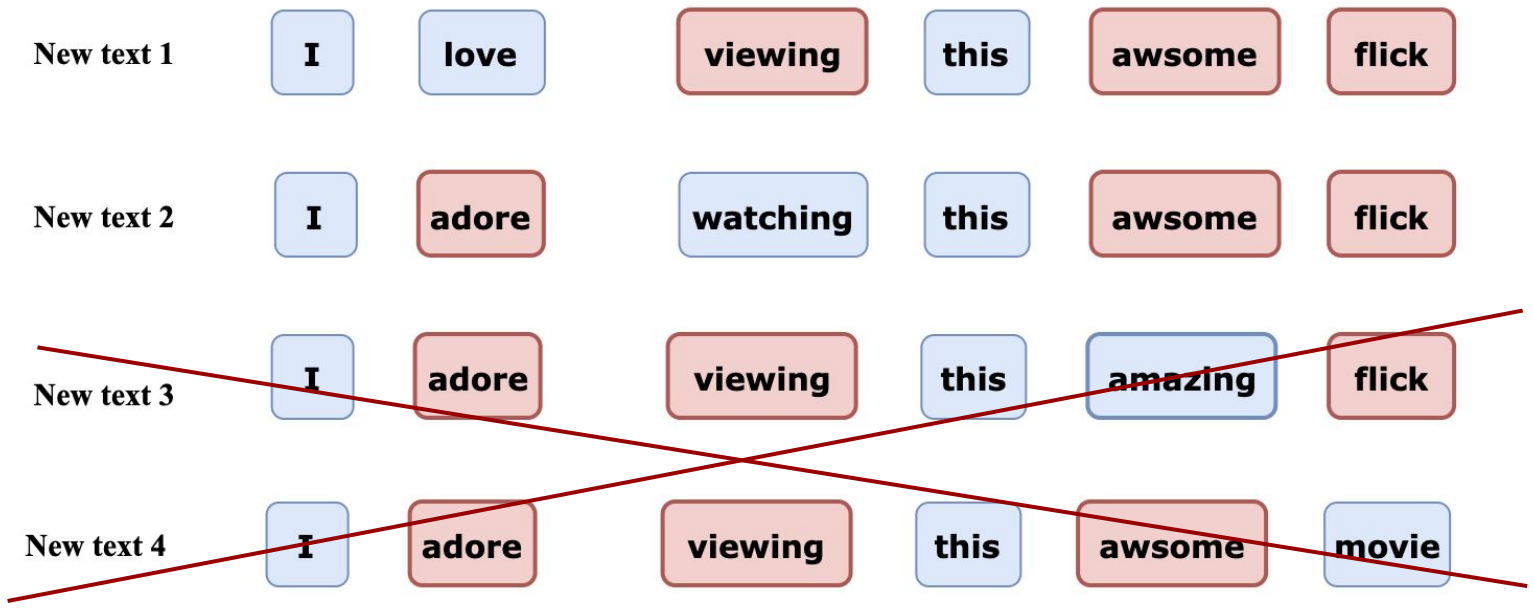
- It replaces each word with its original counterpart.



Step 2: Search Space Reduction

This steps reduces the number of substituted words in the initial adversarial example by replacing back the substituted words with their original counterparts.

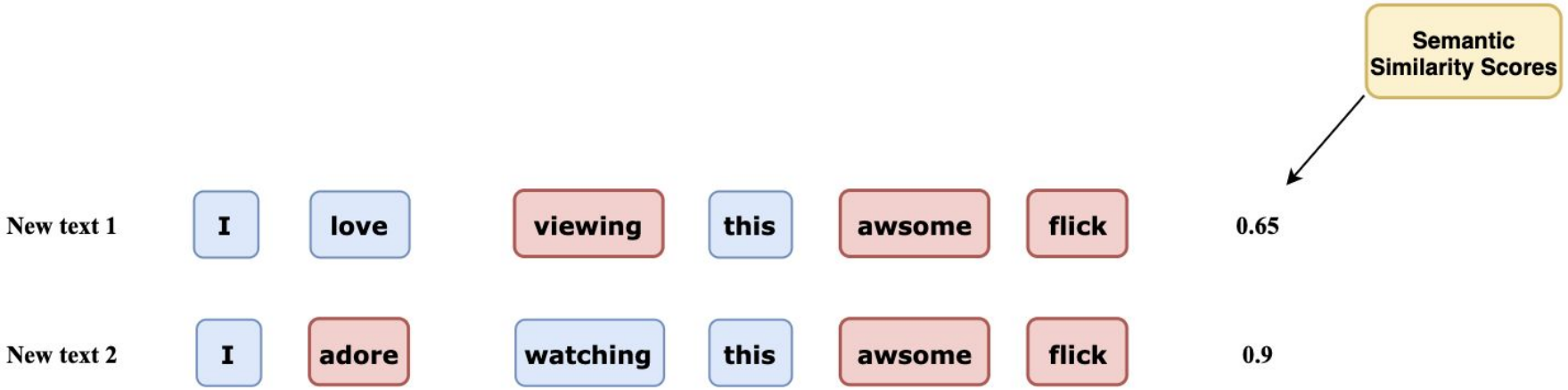
- ❑ It replaces each word with its original counterpart.
- ❑ Filters generates texts which do not satisfy the adversarial criteria.



Step 2: Search Space Reduction

This steps reduces the number of substituted words in the initial adversarial example by replacing back the substituted words with their original counterparts.

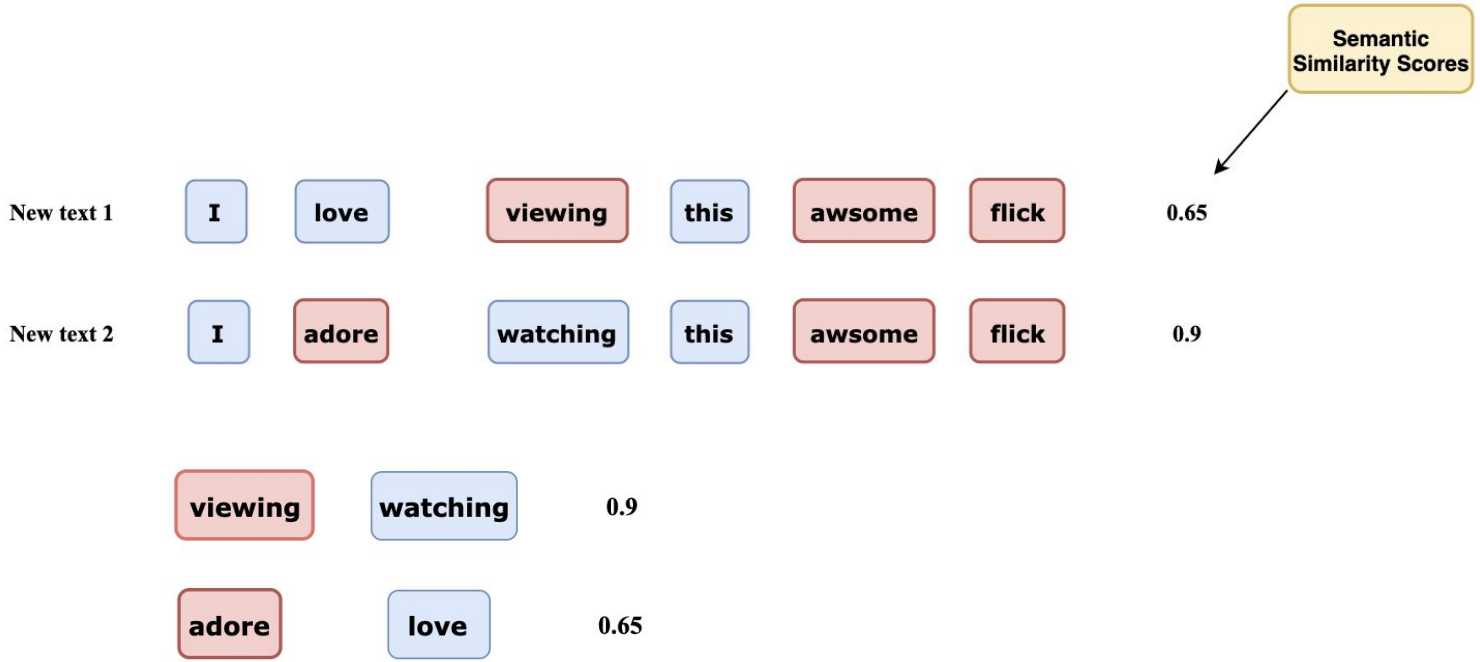
- ❏ It replaces each word with its original counterpart.
- ❏ Filters generates texts which do not satisfy the adversarial criteria.
- ❏ It scores each replacement based on the semantic similarity with the original input.



Step 2: Search Space Reduction

This steps reduces the number of substituted words in the initial adversarial example by replacing back the substituted words with their original counterparts.

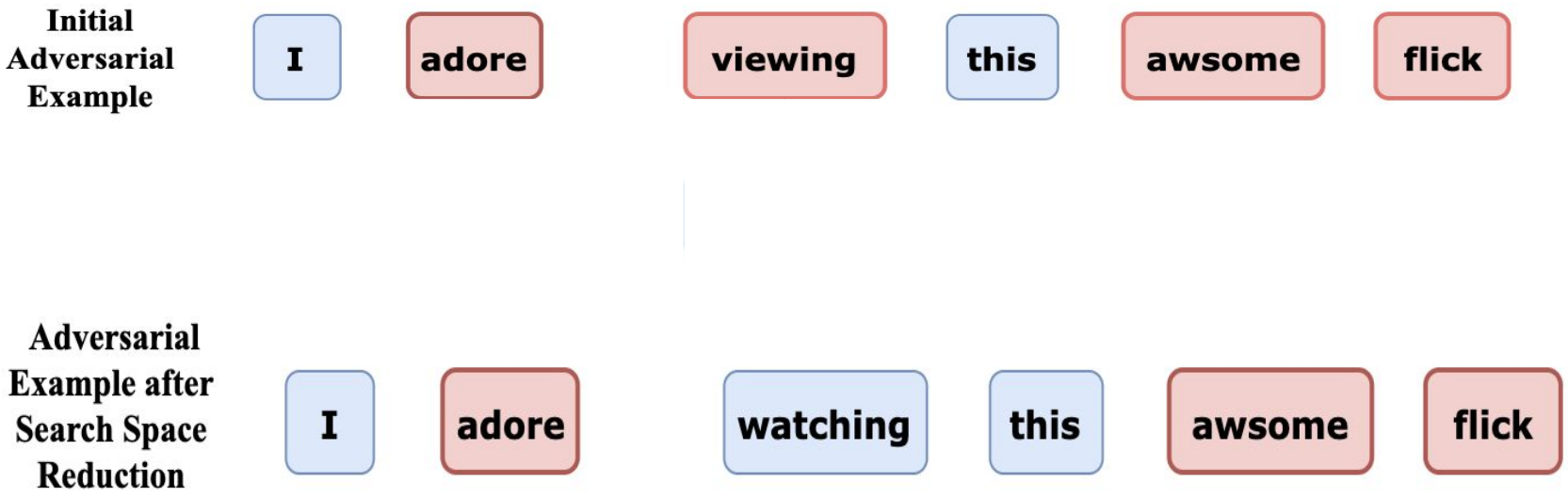
- ❑ It replaces each word with its original counterpart.
- ❑ Filters generates texts which do not satisfy the adversarial criteria.
- ❑ It scores each replacement based on the semantic similarity with the original input.
- ❑ Sort the replacements in descending order based on the above score.



Step 2: Search Space Reduction

This steps reduces the number of substituted words in the initial adversarial example by replacing back the substituted words with their original counterparts.

- ❏ It replaces each word with its original counterpart.
- ❏ Filters generates texts which do not satisfy the adversarial criteria.
- ❏ It scores each replacement based on the semantic similarity with the original input.
- ❏ Sort the replacements in descending order based on the above score.
- ❏ Replaces the synonyms in the initial adversarial example based on the order defined above until the sample remains adversarial.

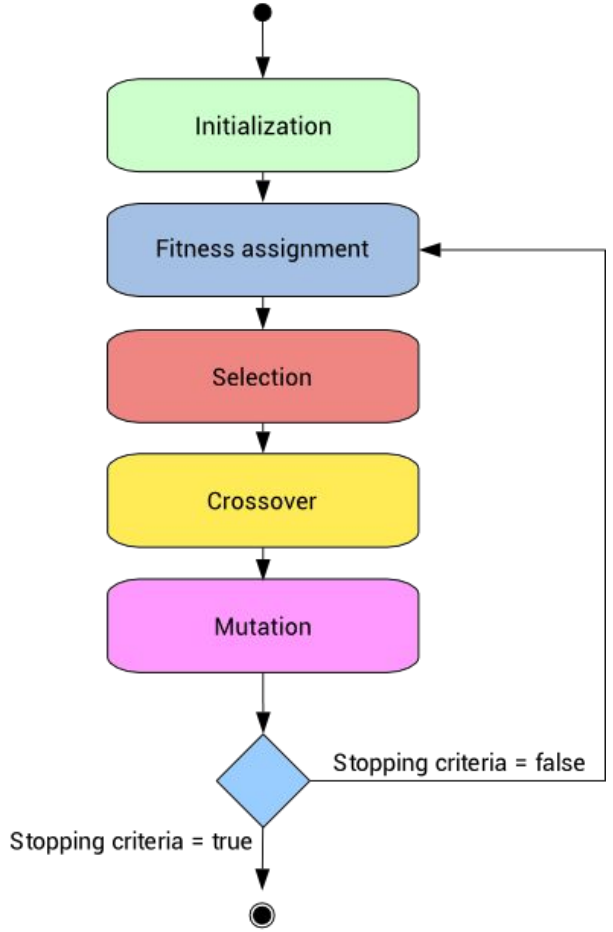


Step 3: Optimization

Genetic Algorithm

1. **Initialisation**: GA starts with an initial set of candidates.
2. **Selection**: Each candidate is evaluated using a fitness function. Two candidates (parents) are selected based upon their fitness values.
3. **Crossover**: The selected parents undergoes crossover to produce the next set of candidates.
4. **Mutation**: The new candidates are mutated to ensures diversity and better exploration of search space.

Steps 2-4 are repeated for a specific number of iterations



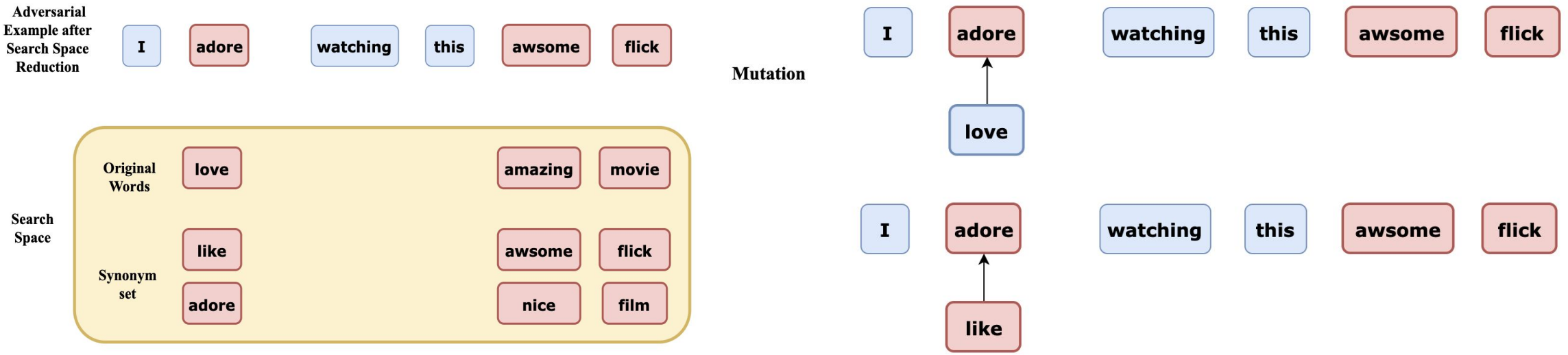
Step 3: Optimization

Population based optimization algorithm

1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.

Mutation

❏ It replaces the substituted index with its original counterpart and with other synonyms from the search space.



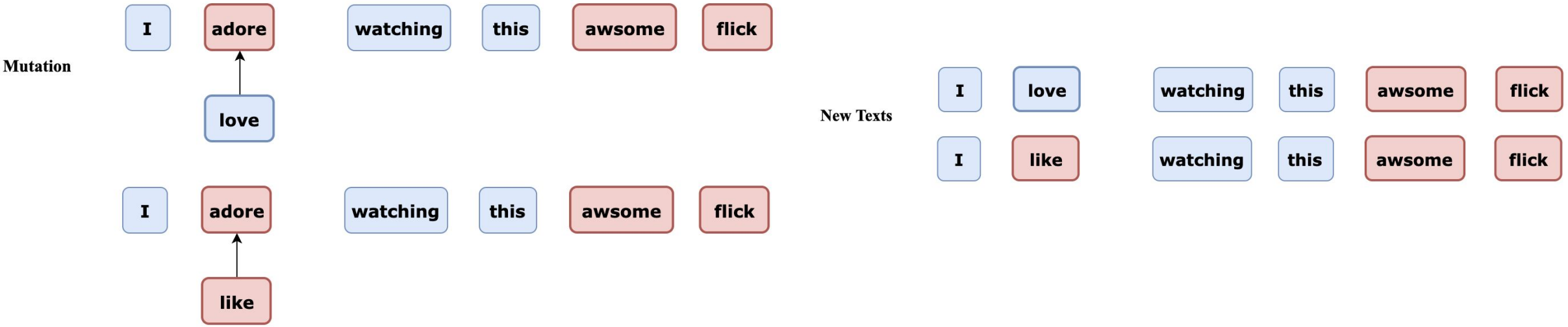
Step 3: Optimization

Population based optimization algorithm

1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.

Mutation

❏ It replaces the substituted index with its original counterpart and with other synonyms from the search space.



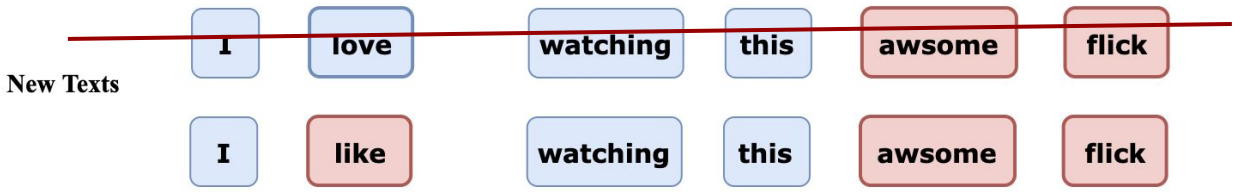
Step 3: Optimization

Population based optimization algorithm

1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.

Mutation

- ❑ It replaces the substituted index with its original counterpart and with other synonyms from the search space.
- ❑ The samples which do not satisfy the adversarial criteria and does not improve the semantic similarity are filtered.
- ❑ From the remaining samples it selects a sample with the maximum semantic similarity with the original input.



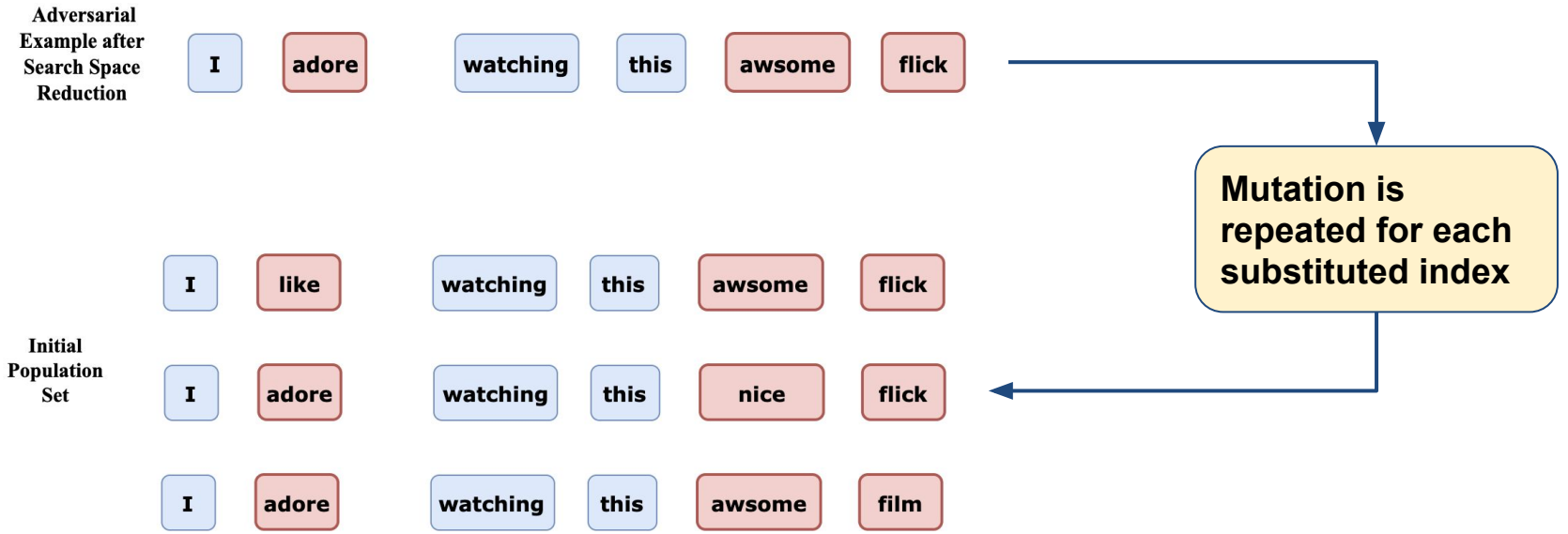
Step 3: Optimization

Population based optimization algorithm

1. **Initial Population:** This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.

Mutation

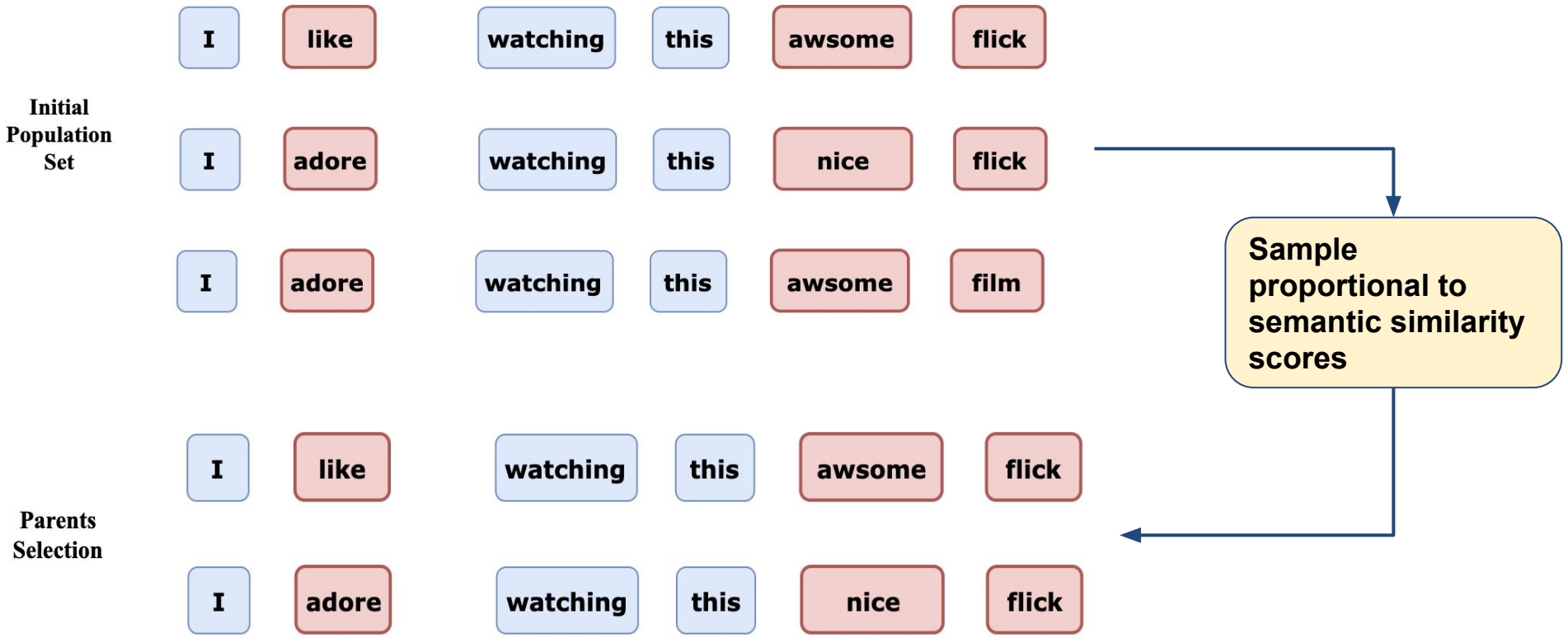
- ❏ It replaces the substituted index with its original counterpart and with other synonyms from the search space.
- ❏ The samples which do not satisfy the adversarial criteria and does not improve the semantic similarity are filtered.
- ❏ From the remaining samples it selects a sample with the maximum semantic similarity with the original input.



Step 3: Optimization

Population based optimization algorithm

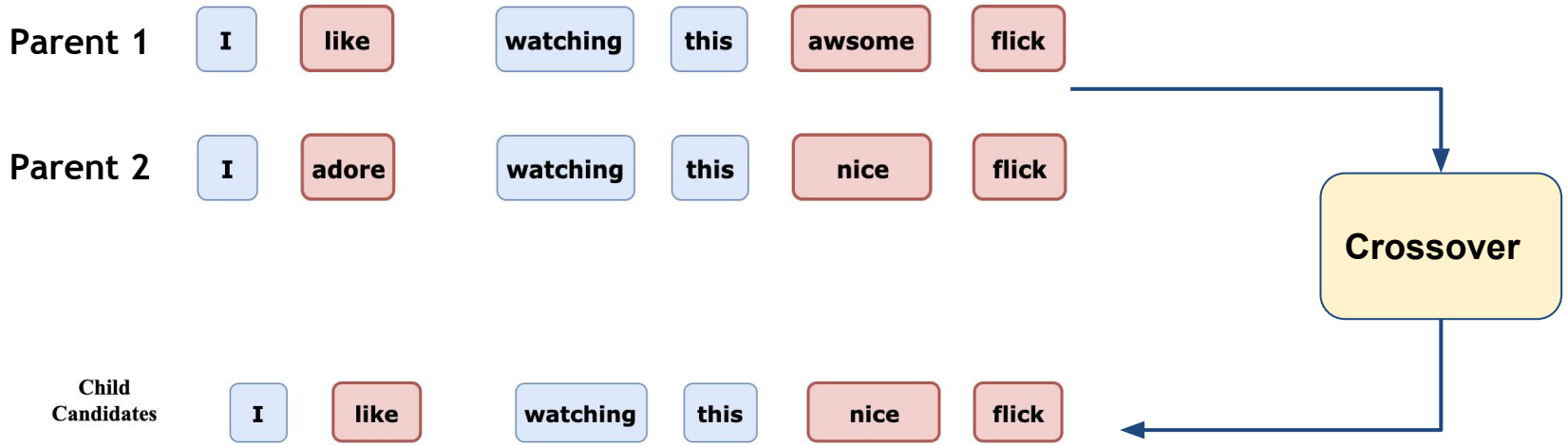
1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.
2. **Selection**: This step samples two parents texts with probability proportional to the semantic similarity scores of the population members.



Step 3: Optimization

Population based optimization algorithm

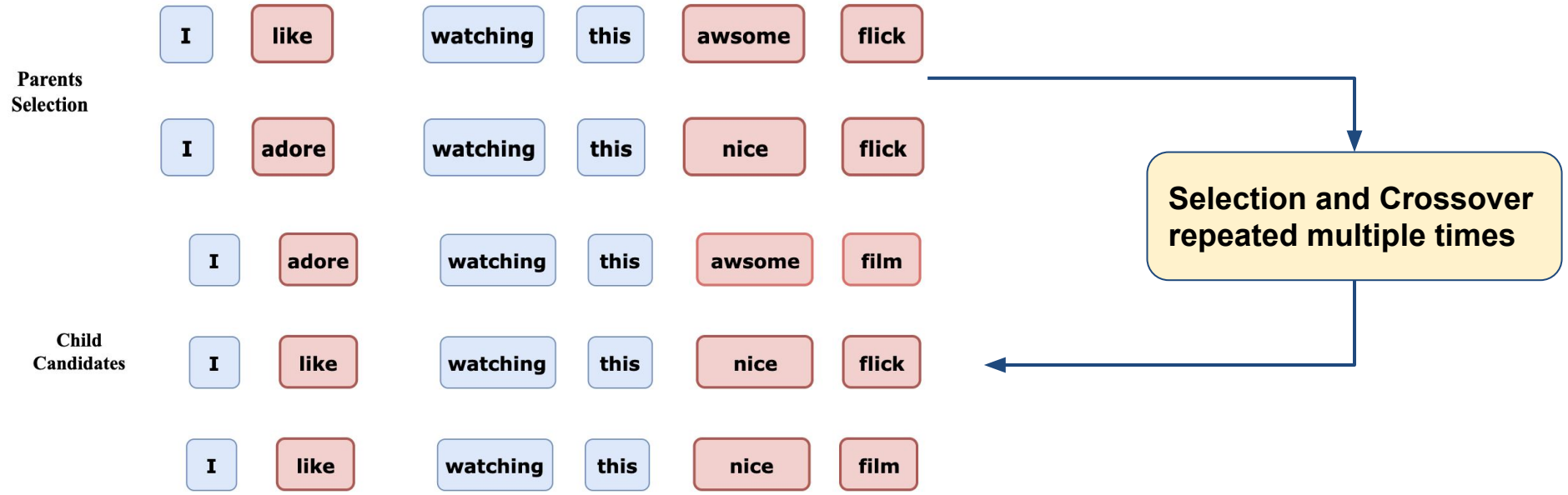
1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.
2. **Selection**: This step samples two parents texts with probability proportional to the semantic similarity scores of the population members.
3. **Crossover**: The parents will go crossover to generate child candidate texts.



Step 3: Optimization

Population based optimization algorithm

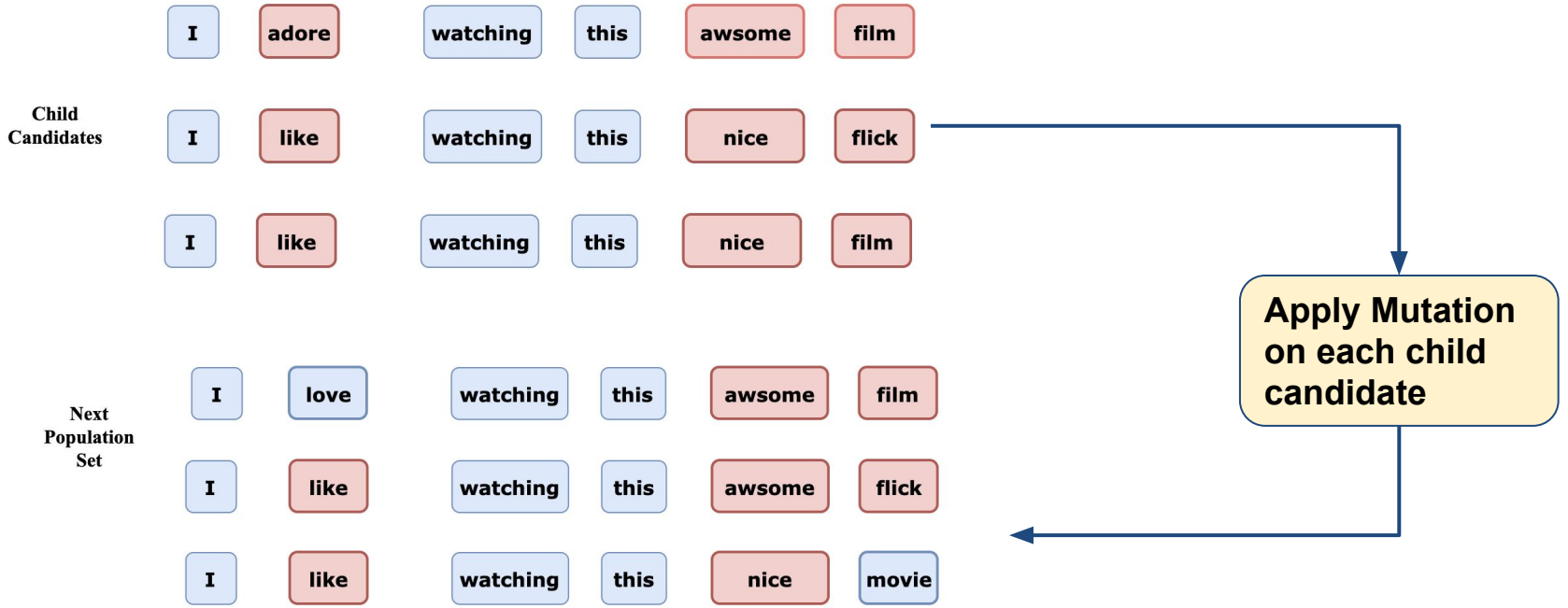
1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.
2. **Selection**: This step samples two parents texts with probability proportional to the semantic similarity scores of the population members.
3. **Crossover**: The parents will go crossover to generate child candidate texts. Selection and Crossover will be repeated multiple times till the number of generated children is equal to population size.



Step 3: Optimization

Population based optimization algorithm

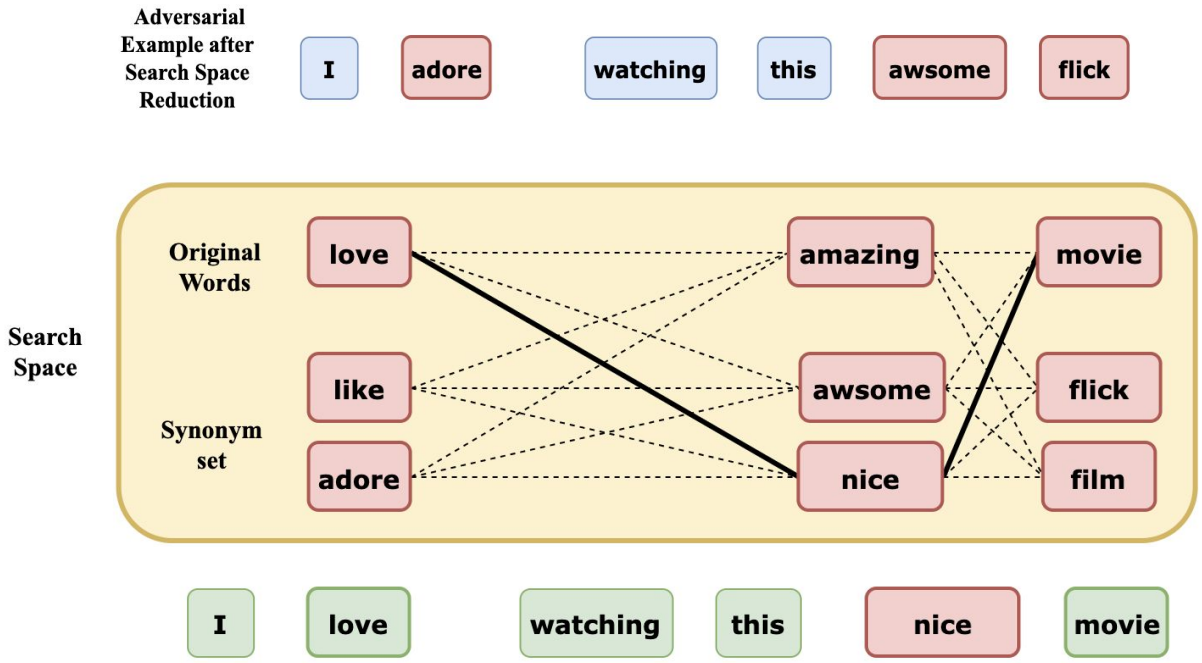
1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.
2. **Selection**: This step samples two parents texts with probability proportional to the semantic similarity scores of the population members.
3. **Crossover**: The parents will go crossover to generate child candidate texts. Crossover will be repeated multiple times till the number of generated children is equal to population size.
4. **Mutation**: The generated children will undergo mutation to generate the next set of population members.



Step 3: Optimization

Population based optimization algorithm

1. **Initial Population**: This steps **mutates** the adversarial sample (obtained after search space reduction) to generate an initial population.
2. **Selection**: This step samples two parents texts with probability proportional to the semantic similarity scores of the population members.
3. **Crossover**: The parents will go crossover to generate child candidate texts. Crossover will be repeated multiple times till the number of generated children is equal to population size.
4. **Mutation**: The generated children will undergo mutation to generate the next set of population members.



Experiments

Natural Language Processing Tasks:

- ❏ Text Classification
- ❏ Natural Language Inference

Datasets:

- ❏ MR (Movie Reviews)
- ❏ IMDB dataset
- ❏ Yelp Reviews
- ❏ AG News
- ❏ Yahoo Answers
- ❏ SNLI
- ❏ MNLI

Target Models:

- ❏ BERT base uncased
- ❏ WordLSTM
- ❏ WordCNN
- ❏ InferSent
- ❏ ESIM

Evaluation Metrics:

- ❏ Attack Success rate
- ❏ Word Perturbation rate
- ❏ Grammatical Correctness
- ❏ Human Evaluation

Results

Dataset	Attack	BERT				WordLSTM				WordCNN			
		Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%
MR	TF	86.00	11.5	16.7	1.26	80.7	3.1	14.90	1.04	78.00	2.8	14.3	**1.27**
	Ours		**7.4**	**10.7**	**1.04**		**2.8**	**12.2**	**0.93**		**2.5**	**11.9**	1.30
IMDB	TF	90.00	13.6	6.10	0.5	89.8	0.3	5.1	0.53	89.20	**0.0**	3.50	0.40
	Ours		**1.1**	**3.13**	**0.36**		**0.2**	**2.9**	**0.27**		**0.0**	**2.8**	**0.37**
Yelp	TF	96.50	6.6	13.9	1.01	95.00	2.1	10.76	1.07	93.80	**1.1**	8.30	0.84
	Ours		**5.2**	**6.37**	**0.62**		**3.2**	**6.7**	**0.62**		**1.1**	**6.44**	**0.78**
AG	TF	94.20	12.5	22.0	1.58	91.30	**3.8**	18.6	1.35	91.5	1.5	15.0	0.91
	Ours		**5.8**	**12.2**	**0.74**		4.1	**12.9**	**0.83**		**1.0**	**10.2**	**0.90**
Yahoo	TF	79.10	18.2	17.7	1.72	73.7	16.6	18.41	0.94	71.1	9.2	15.0	0.80
	Ours		**8.0**	**4.5**	**0.44**		**4.2**	**6.3**	**0.67**		**2.4**	**6.1**	**0.65**
Dataset	**Attack**	**BERT**				**InferSent**				**ESIM**			
		Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%
SNLI	TF	89.1	**4.0**	18.5	9.7	84.0	3.5	18.0	7.7	86.0	5.1	18.1	8.6
	Ours		5.2	**16.7**	**3.70**		**4.1**	**18.2**	**3.70**		**4.1**	**17.2**	**3.62**
MNLI	TF	85.1	9.6	15.4	7.7	70.9	6.7	14.0	4.7	77.9	7.7	14.5	**1.6**
	Ours		**5.2**	**13.5**	**2.79**		**5.1**	**13.2**	**2.98**		**5.9**	**13.1**	2.76
MNLIm	TF	82.1	8.3	14.6	7.3	69.6	6.9	14.6	3.6	75.8	7.3	14.6	2.6
	Ours		**4.1**	**12.71**	**2.56**		**4.5**	**12.8**	**3.1**		**4.0**	**12.6**	**2.4**

Table 2: Comparison with TextFooler (TF). Orig.% is the original accuracy, Acc.% is the after attack accuracy, Pert.% is the average perturbation rate and I% is the average grammatical error increase rate. Mnlim is the mis-matched version of MNLI.

Results

Dataset	Model	Attack	Succ.%	Pert.%
IMDB	**WordLSTM**	TextBugger	86.7	6.9
		Genetic	97.0	14.7
		PSO	**100.0**	3.71
		Ours	99.8	**2.9**
	WordCNN	PWWS	95.5	3.8
		DeepRL	79.4	-
		Ours	**100.0**	**2.8**
	BERT	PSO	98.7	3.69
		Ours	**98.9**	**3.13**
SNLI	**Infersent**	PSO	73.4	**11.7**
		Genetic	70.0	23.0
		GANs	69.6	-
		Ours	**96.6**	17.7
	BERT	PSO	78.9	**11.7**
		Ours	**94.8**	16.7
AG	**WordCNN**	PWWS	43.3	16.7
		Ours	**99.0**	**10.2**
Yahoo	**WordCNN**	PWWS	42.3	25.4
		Ours	**97.6**	**6.1**

Table 3: Comparison with other baselines. Succ.% is attack success rate and Pert.% is average word perturbation rate.

Results

Examples	Prediction
Highly [Exceedingly] watchable stuff.	**Positive → Negative**
It's weird, wonderful, and not **necessarily [definitely]** for kids.	**Negative → Positive**.
Could i use both Avast and Avg to protect my **computer [machinery]**? I recommend the free version of Avg antivirus for home users.	**Technology → Music.**
Premise: Larger ski resorts are 90 minutes away. **Hypothesis**: If we travel for 90 minutes, we could arrive at larger **ski [skating]** resorts.	**Entailment → Neutral.**
Premise: A portion of the nation's income is saved by allowing for capital investment. **Hypothesis**: The nation's income is divided into portions **[fractions]**.	**Entailment → Neutral.**

Table 4: Adversarial samples generated on BERT. The actual word is bold and substituted word are bold and in square brackets.

Evaluation criteria	IMDB	SNLI
Grammatical Correctness	4.44	4.130
Semantic Similarity	0.93	0.896

Table 8: Demonstrates scores given by evaluators

Ablation Study

Ablation:

- ❑ No Search Space Reduction and Genetic Optimization
- ❑ No Search Space Reduction
- ❑ No Genetic Optimization
- ❑ With both Search Space Reduction and Genetic Optimization

Ablation Study	Metric	IMDB	Yelp	SNLI
no SSR and GA	Pert%	20.1	24.3	34.6
	Sim	0.6	0.57	0.22
only GA	Pert%	4.2	7.2	18.0
	Sim	0.81	0.74	0.37
only SSR	Pert%	6.0	9.7	21.0
	Sim	0.80	0.74	0.26
both SSR and GA	Pert%	**3.1**	**6.7**	**16.7**
	Sim	**0.89**	**0.80**	**0.45**

Table 5: Importance of the search space reduction (SSR) and GA step. Pert.% is the average perturbation rate and Sim is the average semantic similarity.

Transferability

An adversarial example is called transferable if it's generated against a particular target model but successfully attacks other target models as well. We evaluate the transferability of adversarial attacks generated on IMDB and SNLI datasets.

Model	BERT	W-CNN	W-LSTM
BERT	-	85.0	86.9
W-CNN	84.6	-	79.1
W-LSTM	80.8	73.6	-

Model	BERT	ESIM	Infersent
BERT	-	53.0	38.5
ESIM	54.9	-	38.5
Infersent	67.4	69.5	-

Table 7: Transferability on IMDB (upper half) and SNLI (lower half) datasets. Row i is the model used to generate attacks and column j is the model which was attacked.

Adversarial Training

We generated adversarial examples on 10% samples of IMDB and SNLI datasets and augmented the generated examples to the original training set.

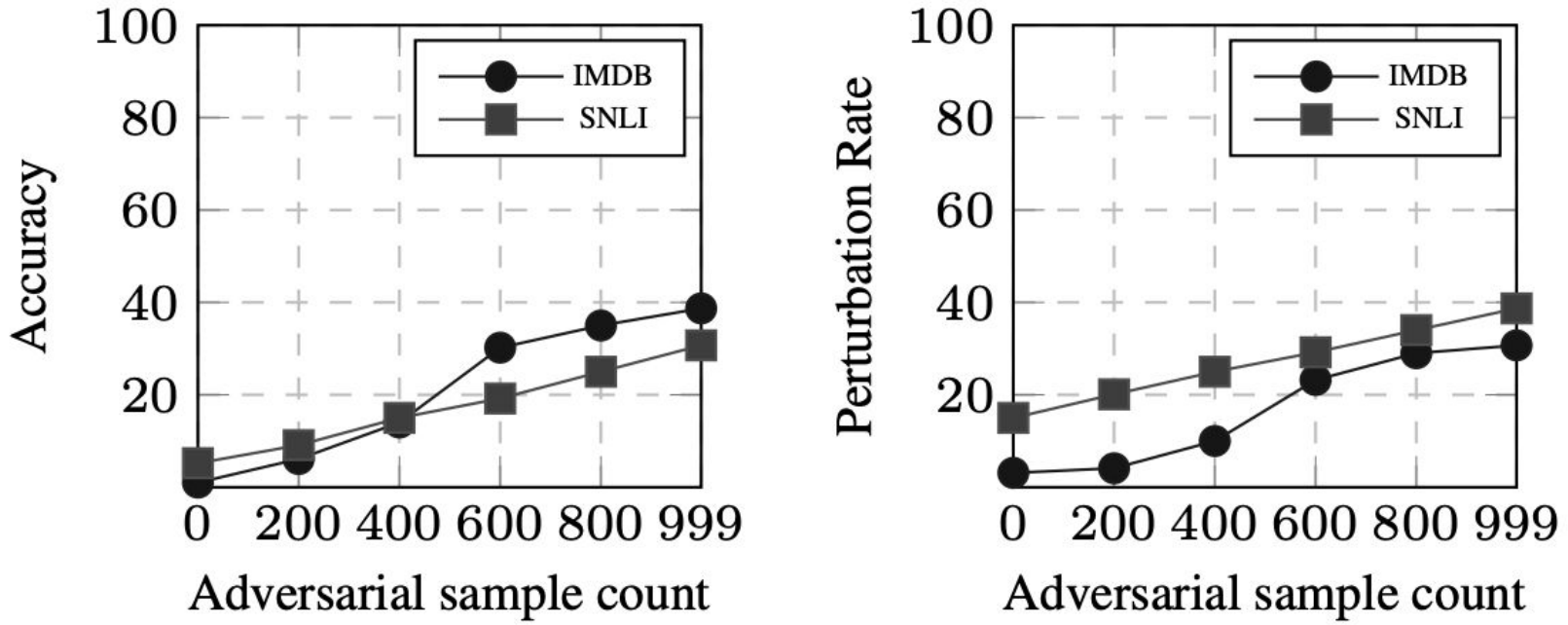


Figure 3: Demonstrates increase in after attack accuracy and perturbation as more adversarial samples are augmented.

Conclusion

In this work, we propose a **novel decision-based attack** that utilizes population-based optimization algorithm to craft **plausible and semantically similar** adversarial examples by o**bserving only the topmost predicted label**.

Our proposed attack does not requires access to:

- ❏ Target model parameters
- ❏ Confidence scores
- ❏ Training data information
- ❏ Substitute models

Extensive experimentation across multiple target models and benchmark datasets on several state of-the-art baselines demonstrate the effectiveness of our proposed attack.

In comparison to prior attack strategies, our attack achieves a higher success rate and lower perturbation rate that too in a highly restricted setting.

Future Work:

- ❏ The proposed attack can be improved to make it more query efficient.
- ❏ The quality of generated adversarial examples can be improved further.

Implementation

- **The arxiv version of our paper is available at**

  https://arxiv.org/abs/2012.14956

- **The code and pretrained models are available at**

  https://github.com/RishabhMaheshwary/hard-label-attack

- **Our attack is also implemented in the TextAttack library** (https://github.com/QData/TextAttack).