Group3: Adversarial Techniques in NLP

Abhishek Krishna Deeksha Arora Preeti Singh
Sambhrant Maurya Shruti Sharma
20111002, 20111017, 20111044, 20111054, 20111061
{krishnacs20, deeksha20, preeti20, samaurya20, shruti20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

May 8, 2021

Abstract

As more and more AI systems are developed, the risk and the need for their security also grows manifold. Machine Learning systems are vulnerable to numerous real-life mischiefs. Recent attack methods on these models have exposed their vulnerability to specific adversarial inputs. Research in Adversarial attacks in NLP has seen a boom in the past four years. We propose a novel idea of employing the recently developed FastText Embeddings(1) for generating robust word replacements. We compare different embeddings (FastText/Word2Vec and GloVe), and our results show that FastText embedding works better than the other two on most of the datasets and models in terms of attack success rate. We call our attack “FastAttack", and we compare our attack against existing benchmarks. We also experiment with Misspellings Oblivious Embeddings(2) and report our findings.

1 Introduction

With the recent advancements in technology, much development has been done for deploying Machine Learning and Deep Learning models in relevant fields. However, all these systems have exposed their over-confident, and foolish attitude towards adversarial (manipulated) input data (3)(4) which highlights the lack of perception in the algorithms.
An *adversarial example* involves a slight change in the input of a machine/deep learning model, which causes the model to predict incorrect labels. It is simpler to generate such examples in image processing/computer vision; however, in Natural Language Processing, depending on the context, an adversarial example should be grammatical and semantically similar to the original input. This motivated us to work in this field and explore the underlying ideas of different types of attacks on natural language. Adversarial techniques have applications in Sentence/Text classification, Malware Detection, Machine Comprehension, Machine Translation, Medical Records, Paraphrasing, etc.

Adversarial attacks can be categorized as follows (5)(6):

1. Based on model accessibility, an attack can be a **white-box attack** or a **black-box attack**. The attacker has access to model parameters in the former, but in the latter, the attacker has no information about the model.
2. Attack can be **targeted** or **untargeted**. In targeted attacks, the attacker wants the model to predict a specific target label, whereas in untargeted attacks, the attacker wants the model to predict any label that is different from the true label.
3. Based on Granularity, an attack can be at **character-level**, **word-level**, or **sentence-level**.
4. Attack can be **single modal** like on Dialogue Systems or **cross modal** like on Speech Recognition Systems.
5. Based on the type of Deep Neural Network being attacked, the attack can be on CNNs, RNNs, Hybrid models, etc.

Problem Statement:

- To devise an efficient and fast attack recipe and compare our new attack with existing benchmark attacks.
- Develop an embedding which generates efficient word replacements.

In this mini-project, we devise a new replacement strategy using FastText and perform experiments on the TextAttack framework. We show that FastText replacement works better than Word2Vec and GloVe embeddings. Using our new replacement strategy, the results show that we are able to beat most of the existing attack recipes in literature in terms of attack success rate, perturbation percentage and attack time. Section 2 of the report describes the existing work done on Adversarial attacks in NLP. In section 3, we have described our proposed idea in detail. Section 4 contains details about the datasets that were used in the mini-project. Section 5 displays experiments on different datasets and models, and the corresponding results are shown in Section 6. In the last, we describe error analysis, conclusion, and future improvements. Source-Code is available at: `https://github.com/deekshaarora05/FastAttack`

2 Related Work

In literature, researchers have come up with myriad attack strategies on NLP applications and aimed to improve the robustness and generalization of these systems. However, the catch is to generate adversarial examples in a cost-sensitive and speedy manner. There have been proposed approaches based on gradients (7) (8) of training examples to craft adversarial sequences. A few modifications, like taking into account the direction of derivatives (9), performing multi-modal attacks (10) using iterative optimization based techniques on regularized objectives (11), generating adversarial examples in semantic space (12) using deep generative models like GANs, VAEs etc., or using MCMC-based sampling methods (13) for example generation have been suggested in previous research. Optimisation based frameworks like TextBugger (14) and Seq2Sick (15) have been developed to evaluate the robustness of models.

An adversarial example should be such that it goes unperceived by humans and still is able to fool the model. To satisfy this constraint during adversarial example generation various human-in-the-loop approaches (16) (17) (18) (19) have been proposed. This strategy aims to create more complex and diverse failure patterns.

The generated sequences should be semantically similar to the original sequence and shouldn't change the meaning of background context. In this regard, attack strategies like limiting the direction of perturbations in input embedding space (20), computing scores (21) while performing modifications, substitution using synonyms of input text (22) (23) (24) (25), using inflectional morphology of words (26), appending grammatical/non-grammatical sentences to input (27) (28) (29) have been explored.

Sentence paraphrasing (30) has delivered satisfactory success in attacking target models. To launch this type of attack, trained SCPNs (31) (32) and transformer-based models (33) have been used to paraphrase the sentence. Datasets (34) containing paraphrased and non-paraphrased pairs are also curated.

Some other works have leveraged performance of BERT-based (35) (36) Masked Language Models or employed Occlusion and Language Models (37) to perform efficient attacks. (38) performed population based optimization via genetic algorithms to generate syntactically and semantically similar adversarial examples. (39) iteratively approximated the decision boundary of Deep Neural Networks to generate adversarial text. (40) used conditional sequential text generation model to generate perturbed comments. Morris et.al.(2020)(41) introduced a new python open-source framework, TextAttack. TextAttack is designed to benchmark and compare different NLP attacks from literature on various datasets. An attack can be composed using a goal function, set of constraints, transformation and search method. Yong et.al.(2020) (42) tried to find out which search method should be used to generate adversarial examples. On Yelp, MR and SNLI datasets, results show that in terms of success rate, Beam Search and Particle Swarm Optimization have leading scores and in terms of time constraint, Greedy-WIR is the preferable method. Recently,Modi et.al (2021)(37) have employed the idea of Occlusion and Language Models(OLM) to perform efficient word ranking where OLM is used to generate the OLM relevance score by sampling some candidate instances for a word and then replacing that word.

3 Proposed Idea

Previously, Word2Vec and GLoVe have been the go-to word-representation construction methods for majority of NLP tasks. Infact, TextAttack, the base framework for our experiments, uses 200-dim GLoVe embeddings in default setting. So we propose to use more advanced embeddings like FastText (1) and Misspelling Oblivious Word Embeddings (MOE) (43) that have a better provision of handling OOV(Out-Of-Vocabulary) words. Both of these methods have been explored by Facebook Research. In our attack, named FastAttack, we leverage FastText to design a better and faster attack methodology. Based on the results obtained from experiments described in section 5, we find that GreedyWordSwapWIR("weighted saliency") gives desirable results using FastText embedding and provides a good balance between performance and speed. The four components defining our attack are:

- **Goal Function: Untargeted Classification -** Aims to minimize the score of the correct label until it is no longer the predicted label.
- **Constraints: Stopword Modification + Repeat Modification -** Prevents modification of stop words and words which already have been modified.
- **Transformation:** Replace the chosen word $w \in W$ with its closest embedded word w' in the FastText embedding space under the following constraints:
  1. w' isn't a substring of w.
  2. w' isn't a stopword. (NLTK Stopwords)
  3. In a sentence, word w should be modified only once.
  4. The case of replaced word should be preserved.
- **Search Method: GreedyWordSwapWIR (22) -** GreedyWordSwapWIR with weighted saliency is used as the search method. It uses word importance ranking and greedily chooses the word to be perturbed from a list of possible perturbations.

One of the main limitation is that we trained embeddings on 'text8' dataset which may not contain all the possible words of the English language. For some of the cases, trained embeddings aren't able to find the correct word replacement resulting in less attack success rate and accuracy. We compare our attack with most of the existing benchmarks but still, there is a possibility to compare with rest of the existing attacks from the literature.

4 Dataset/Corpus

For training FastText, Word2Vec and GloVe embeddings, we used *text8* dataset made available by Gensim for unstructured text processing. This dataset is publicly available at text8. MOE embedding is trained on moe_misspellings_train dataset which is available here. To carry out the experiments, we used *AG-News, Glue(subset- QQP, SST2), IMDB, Rotten Tomatoes and Yelp Polarity* datasets made available by Hugging Face.

Dataset	Classes	Train	Test	Avg. Length
AG-News	4	120K	7600	245.12
IMDB	2	25K	25K	245.12
Yelp	2	560K	38K	132.33
Rotten Tomatoes	2	8530	1066	-
Glue/SST-2	2	67K	1821	-
Glue/QQP	2	363K	390K	-

Table 1: Statistics of Datasets. Avg. Length is the average number of words in the test set.

5 Experiments

We trained FastText, Word2Vec and GloVe embeddings on *text8* dataset, consisting of first 100,000,000 bytes of plain text from Wikipedia. To learn the underlying word representations for each word, Skip-gram architecture is used instead of CBOW architecture. The hyperparameters for embeddings are mentioned in Table 2. All the experiments are carried out on Google Colab platform.

Embedding	Model Type	Vector Size	Window Size	Min count	Learning Rate	#Epochs	#Threads	Training Time(s)
FastText	Skipgram	100	5	5	0.025	5	3	932
Word2Vec	Skipgram	100	5	5	0.025	5	3	670
GloVe	-	100	5	1	0.05	5	3	628
MOE	Skipgram	100	2	5	0.05	5	3	612

Table 2: Hyperparameters for Different Embeddings

The experiments are carried out using three search methods: *Greedy Search* (GS), *Greedy Word Swap Word Importance Ranking* (GWS) and *Particle Swarm Optimization* (PSO). To carry out the attack, we used StopWord Modification and Repeat Modification constraint along with Untargeted Classification as the goal function.

We experimented with different benchmark datasets for text classification: IMDB, AG News, Yelp Polarity, Rotten Tomatoes and SST-2. The statistics of the dataset are shown in Table 1.
We evaluated the performance of our proposed attack using transformer-based SOTA text classifiers: BERT [Devlin et al.,2019 (44)], DistilBERT [Sanh et al., 2020 (45)], ALBERT [Lan et al., 2020 (46)] and RoBERTa [Liu et al., 2019 (47)].

The attack results for FastText, Word2Vec and GloVe emebeddings are shown in Table 3 and Table 4.

Dataset	**Embedding**	**Search**	**Yelp-Polarity**			
		Method	**Original Acc.**	**Attack Acc.**	**Success Rate**	**Perturbed %**
Albert-Base	FastText	GS	100.0%	**8.0%**	**92.0%**	10.66%
		GWS		**28.0%**	**72.0%**	12.89%
		PSO		**32.0%**	**68.0%**	8.15%
	Word2Vec	GS		44.0%	56.0%	9.06%
		GWS		52.0%	48.0%	14.63%
		PSO		56.0%	44.0%	**7.6%**
	GloVe	GS		28.0%	72.0%	**5.63%**
		GWS		36.0%	64.0%	**10.06%**
		PSO		36.0%	64.0%	7.75%
BERT-uncased	FastText	GS	100.0%	**0.0%**	**100.0%**	9.16%
		GWS		**32.0%**	**68.0%**	11.52%
		PSO		**20.0%**	**80.0%**	10.31%
	Word2Vec	GS		32.0%	68.0%	9.58%
		GWS		48.0%	52.0%	11.11%
		PSO		48.0%	52.0%	9.33%
	GloVe	GS		12.0%	88.0%	**8.78%**
		GWS		40.0%	60.0%	**10.1%**
		PSO		32.0%	68.0%	**7.58%**

Table 3: Comparison between word embeddings uisng different search strategies on Yelp-polarity dataset for fine-tuned BERT and ALBERT.

From Table 3 and Table 4, it is clear that attacks using FastText embedding have higher success rate as compared to attacks carried out using Word2Vec or GloVe embeddings. In some cases like for Bert-uncased (on IMDB and Yelp Polarity dataset), Distilbert-uncased (on IMDB dataset) and Albert-base-v2 (on IMDB dataset) the attack success rate is as high as 100%. Based on the results obtained from experiments of different search methods, constraint and embedding, we found that approaches like Particle Swarm Optimization and Beam Search with Beam width>4 are very efficient but quite slow during attack. GreedyWordSwap is very fast but gives poor performance. We found GreedyWordSwapWIR(“weighted saliency") to be a good compromise between performance and speed and hence decided to use it as the search method in our attack.

6 Results

In TextAttack (48), the existing attacks are benchmarked using text recipe. To evaluate the performance of our attack, FastAttack, we compare the attack results with the following existing attack recipe’s: BAE (49), BERT-Attack (50), DeepWordBug (21), Fast-Alzantot (51), Pruthi (52), TextBugger (53) and TextFooler (24). We used the IMDB and Yelp Polarity dataset with BERT and ALBERT models. For IMDB dataset, the model is queries for 10 examples and for Yelp Polarity dataset the model is queried for 25 examples. The following evaluation metrics are used:

1. **Attack Acc.:** Accuracy of the model under attack. Lower the better.
2. **Success Rate:** Represents success rate of the attack and is equal to number of successful attacks divided by total number of attempted attacks. Higher the better.

Model	Embedding	Search	IMDB				AG NEWS			
		Method	Original Acc.	Attacked Acc.	Success Rate	Perturbed %	Original Acc.	Attacked Acc.	Success Rate	Perturbed %
Bert-uncased	FastText	GS	90.91%	**0.0%**	**100.0%**	5.64%	96.15%	**46.15%**	**52.0%**	19.98%
		GWS		**27.27%**	**70.0%**	8.56%		**61.54%**	**36.0%**	19.19%
		PSO		**27.27%**	**70.0%**	**2.49%**		**53.85%**	**44.0%**	21.67%
	Word2Vec	GS		**0.0%**	**100.0%**	5.31%		88.46%	8.0%	**3.65%**
		GWS		45.45%	50.0%	11.07%		88.46%	8.0%	**7.29%**
		PSO		**27.27%**	**70.0%**	5.12%		88.46%	8.0%	**3.65%**
	GloVe	GS		9.09%	90.0%	**3.7%**		84.62%	12.0%	8.31%
		GWS		**27.27%**	**70.0%**	**7.38%**		84.62%	12.0%	10.74%
		PSO		36.36%	60.0%	3.69%		84.62%	12.0%	12.23%
RoBERTa	FastText	GS	88.89%	22.22%	75.0%	5.93%	100.0%	**50.0%**	**48.0%**	22.71%
		GWS		**33.33%**	**62.5%**	**6.89%**		**65.38%**	**32.0%**	18.36%
		PSO		**44.44%**	**50.0%**	**1.35%**		61.54%	**36.0%**	17.71%
	Word2Vec	GS		22.22%	75.0%	**4.75%**		84.62%	12.0%	**10.71%**
		GWS		**33.33%**	**62.5%**	7.63%		88.46%	8.0%	16.45%
		PSO		**44.44%**	**50.0%**	2.71%		84.62%	12.0%	17.57%
	GloVe	GS		**11.11%**	**87.50%**	11.40%		84.62%	12.0%	13.76%
		GWS		**33.33%**	**62.50%**	9.71%		84.62%	12.0%	21.11%
		PSO		55.56%	37.50%	5.88%		84.62%	12.0%	19.15%
DistilBERT-uncased	FastText	GS	100.0%	**0.0%**	**100.0%**	5.97%	92.59%	**70.37%**	**24.0%**	20.38%
		GWS		**22.22%**	**77.78%**	6.15%		**77.78%**	**16.0%**	13.4%
		PSO		22.22%	77.78%	**3.41%**		**74.07%**	**20.0%**	15.43%
	Word2Vec	GS		**0.0%**	**100.0%**	4.8%		85.19%	8.0%	**3.65%**
		GWS		33.33%	66.67%	10.51%		85.19%	8.0%	**3.65%**
		PSO		33.33%	66.67%	3.65%		85.19%	8.0%	**3.65%**
	GloVe	GS		**11.11%**	88.89%	**2.55%**		81.48%	12.0%	10.27%
		GWS		**22.22%**	**77.78%**	**5.3%**		85.19%	8.0%	**3.65%**
		PSO		**11.11%**	**88.89%**	3.93%		81.48%	12.0%	12.23%
Albert-base-v2	FastText	GS	100.0%	**0.0%**	**100.0%**	**2.32%**	92.59%	**40.74%**	**56.0%**	20.39%
		GWS		**11.11%**	**88.89%**	**4.26%**		**59.26%**	**36.0%**	20.97%
		PSO		22.22%	77.78%	**2.78%**		**48.15%**	**48.0%**	17.01%
	Word2Vec	GS		**0.0%**	**100.0%**	3.41%		85.19%	8.0%	**7.11%**
		GWS		**11.11%**	**88.89%**	6.84%		85.19%	8.0%	**11.27%**
		PSO		**11.11%**	**88.89%**	4.04%		85.19%	8.0%	**7.11%**
	GloVe	GS		11.11%	88.89%	3.54%		74.07%	20.0%	11.63%
		GWS		33.33%	66.67%	5.56%		70.37%	24.0%	15.52%
		PSO		**11.11%**	**88.89%**	4.04%		70.37%	24.0%	14.99%

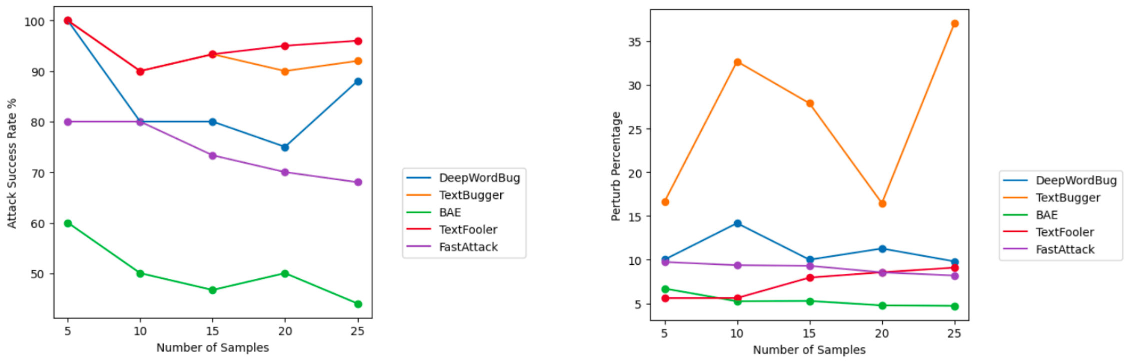
Table 4: Comparison between word embeddings uisng different search strategies on AG News and IMDB datasets for ALBERT, RoBERTa, BERT, and DistilBERT-uncased models.

3. **Perturbed %:** Represents number of words modified under attack divided by total number of words in the input example. Lower the better.

The results of FastAttack, BAE, BERT-Attack, DeepWordBug, Fast-Alzantot, Pruthi, TextBugger and TextFooler attacks on BERT and ALBERT models on IMDB and Yelp Polarity datasets are summarized in table 5. From Figure 1 and Figure 2 it is evident that FastAttack performs better than existing attack recipe.

Attack	Model	IMDB						Yelp Polarity					
		Original Acc.	Attack Acc.	Success Rate	Perturbed %	Queries	Attack Time	Original Acc.	Attack Acc.	Success Rate	Perturbed %	Queries	Attack Time
FastAttack	Bert	92.59%	14.81%	84.0%	5.29%	419.72	268.29	100.0%	16.0%	84.0%	12.51%	238.84	252.68
	AlBert	89.29%	10.71%	88.0%	5.2%	434.92	584.61	100.0%	32.0%	68.0%	8.19%	242.28	146.80
TextBugger	Bert	92.59%	11.11%	88.0%	32.84%	539.12	278.07	100.0%	16.0%	84.0%	34.34%	357.68	176.63
	AlBert	89.29%	3.57%	96.0%	27.25%	490.64	486.14	100.0%	12.0%	88.0%	25.53%	337.72	174.57
DeepWordBug	Bert	92.59%	14.81%	84.0%	4.73%	324.68	111.59	100.0%	28.0%	72.0%	10.03%	247.16	73.39
	AlBert	89.29%	10.71%	88.0%	3.77%	328.84	128.33	100.0%	12.0%	88.0%	9.81%	216.08	72.94
TextFooler	Bert	92.59%	0.0%	100.0%	7.99%	663.64	400.18	100.0%	0.0%	100.0%	10.89%	479.08	237.07
	AlBert	89.29%	0.0%	100.0%	8.35%	674.2	473.95	100.0%	4.0%	96.0%	9.11%	480.92	316.80
BAE	Bert	92.59%	48.15%	48.0%	2.97%	561.52	1153.58	100.0%	56.0%	44.0%	6.44%	314.88	764.20
	AlBert	89.29%	32.14%	64.0%	2.73%	512.32	1051.52	100.0%	56.0%	44.0%	4.72%	287.24	438.28
CAM-RWR	Bert	92.59%	55.56%	40.0%	0.48%	3868.6	2192.64	100.0%	96.0%	4.0%	6.67%	1665.96	848.89
	AlBert	89.29%	60.71%	32.0%	0.53%	3843.64	2493.46	100.0%	92.0%	8.0%	4.28%	1666.0	924.62

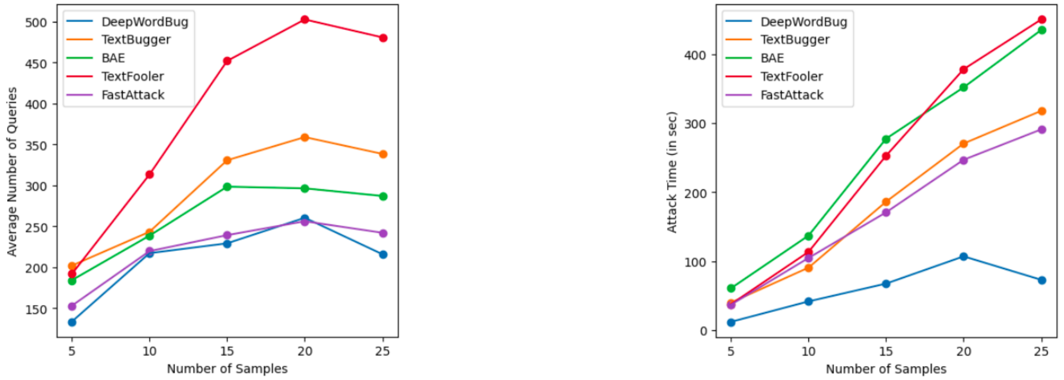
Table 5: Comparison of FastAttack with existing Attack Recipe on Bert-base-uncased and Albert-case-v2 models.



(a) Comparison based on Attack Success Rate

(b) Comparison based on Perturbation Percentage

Figure 1: Comparison of FastAttack with existing Attack Recipe



(a) Comparison based on Avg. #Queries

(b) Comparison based on Attack Time

Figure 2: Comparison of FastAttack with existing Attack Recipe

7 Error Analysis

Neural architectures are designed to train word embeddings and use of these embeddings for other NLP tasks have delivered exemplary results. Popularity of Word2Vec is due to its ability to train large corpus efficiently but it fails in certain cases. FastText improves Word2Vec by introducing subword-level features and giving more attention to OOV(Out-of-Vocabulary) words. Subwords are embedded to create final representation of word w_i. Using min_n and max_n, the minimum and maximum lengths of n-gram, FastText embeds all such possible n-grams of the word. In this way, FastText improves over the Word2Vec skip-gram model which can also be seen in the experiment results. Word2Vec is much like GloVe in the sense that they both treat words as atomic quantity. The effect of this can be seen in experiment results wherein GloVe and Word2Vec deliver similar success rates. MOE is a generalized version of FastText, although Facebook hasn't yet released the training procedure for MOE and neither the trained embeddings, thus in our experiments MOE-based attacks perform worst out of all embeddings, resulting in perturbed sentences that will be perceivable by a human too, thereby violating the principle of adversarial example generation. An example,
Sci/tech (100%) –> World (85%)
Calif. Aims to Limit Farm-Related Smog (AP) AP - Southern California's smog-fighting agency went after emissions of the bovine variety Friday, adopting the nation's first rules to reduce air pollution from dairy cow manure.
Calif. Aims to Limit Farm-Related dog (AP) AP - Southern California's mmafighting agency went after émissions of the bovine variety Friday, adoting the nation's first rules to reduce air polltion from dairy cow mansure.

Fasttext has an overall higher attack success rate as compared to Word2Vec and GloVe. Using FastText embeddings as our baseline for attacks is better since it can handle OOV(Out-of-Vocabulary) words. With GreedySearch strategy, the attack takes more time to generate similar quality attack as compared to other search methods like GreedyWordSwapWIR and ParticleSwarmOptimization. GreedyWordSwapWIR has least attack time. Out of tried datasets (AG_NEWS, IMDB, Rotten Tomatoes Movie Reviews and SST-2), IMDB takes maximum time to be attacked while SST-2 takes least time. On Rotten Tomatoes dataset, it can be seen that GloVe performs better than FastText and Word2Vec. In SST-2, with search method GreedySwapWIR and constraints (RepeatModification + StopwordModification), success rates follows the pattern: $Glove > Word2Vec > FastText$.

Beam Search as a search method could not be tested extensively due to lack of computing resources. With less query budget, Greedy search gave lesser effective results than Greedy-WIR. Change in constraints and transformations affected the attack result more than change in search methods.

Some of the example sentences and comparison of FastAttack with other TextAttack recipes, tested on bert-base-uncased model trained on ag-news dataset,

- **FastAttack**
  Sci/tech (98%) –> World (73%)
  E-mail scam targets police chief Wiltshire Police warns about "phishing" after its fraud squad chief was targeted.
  E-mail scam targets police chief Wiltshire Police warns about "furnishing" after its bribery squad chief was targeted.
- **TextFooler**
  Sci/tech (98%) –> World (54%)
  E-mail scam targets police chief Wiltshire Police warns about "phishing" after its fraud squad chief was targeted.
  E-mail scam targets police chief Wiltshire Police warns about "phishing" after its hoax battalion leiter was targeted.
- **DeepWordBug**
  Sci/tech (98%) –> World (78%)
  E-mail scam targets police chief Wiltshire Police warns about "phishing" after its fraud squad chief was targeted.
  E-mail scam targets police chief Wiltshire Police warns about "phishin" after its fruad squad chief was targeted.
- **BAE**
  Sci/tech (98%) –> [FAILED]

E-mail scam targets police chief Wiltshire Police warns about “phishing" after its fraud squad chief was targeted.

- **BERT-Attack**
  Sci/tech (98%) –> World (82%)
  E-mail scam targets police chief Wiltshire Police warns about “phishing" after its fraud squad chief was targeted.
  E-mail scam targets police chief Wiltshire Police warns about “phitism" after its fraudulent squad chief was targeted.

FastAttack is successfully able to fool the trained model by inserting valid and/or semantically closer words. The percent change in classification category of FastAttack is better than TextFooler. While BAE fails to even attack the concerned sentence, FastAttack is able to generate an adversarial attack by changing “phishing" to “furnishing" and “fraud" to “bribery". Since DeepWordBug utilises replace-delete-add-swap character transformations to generate adversarial examples, it results in invalid word insertion but still is able to fool the model. On the other hand, FastAttack does so in a clever manner and might even skip a human’s eye.

8 Individual Contribution

Roll No.	Name	Contribution
20111002	Abhishek Krishna	Trained FastText and Word2Vec embedding with CBOW architecture; FastAttack Implementation; Evaluated different embeddings on ALBERT models; Evaluated FastAttack against TextBugger and Bae attack recipe; Presentation
20111017	Deeksha Arora	Trained GLoVe embedding; Trained FastText and Word2Vec embeddings with multiword ngram architecture; Evaluated different embeddings on DistilBERT-base-cased, XLNET and ALBERT models; Evaluated FastAttack against Deepwordbug and CAM-RWR (Pruthi) attack recipe; Report
20111044	Preeti Singh	Evaluated different embeddings on DistilBERT-base-uncased model; Report
20111054	Sambrant Maurya	Trained FastText and Word2Vec embedding with skipgram architecture; FastAttack Implementation; Evaluated different embeddings on BERT models; Evaluated FastAttack against TextFooler, BERT and fast-alzantot attack recipe; Presentation
20111061	Shruti Sharma	Trained Misspelling Oblivious Word Embeddings (MOE) with Skip-gram architecture; Evaluated different embeddings on RoBERTa-base; Report

9 Conclusion

In this work, we devised **FastAttack**, a black-box attack which significantly improves the performance over some existing attack methods, in terms of attack success rate, perturb percentage, average number of queries and attack time. FastAttack is a simple and highly potent attack with no complex transformers and no language models and still gives some of the latest attacks a run for their money. It is a fast black-box attack which uses FastText embedding to do replacement only with valid words.

Future Work: The performace of FastAttack can be further improved by using a better search method and developing strategies to find antonyms from the word embeddings instead of selecting the closest word. We would also like to experiment with a better corpus having more words.

References

[1] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2017.

[2] B. Edizel, A. Piktus, P. Bojanowski, R. Ferreira, E. Grave, and F. Silvestri, "Misspelling oblivious word embeddings," *CoRR*, vol. abs/1905.09755, 2019.

[3] J. S. Ian J Goodfellow and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations*, 2015.

[4] I. S. Christian Szegedy, Wojciech Zaremba and J. Bruna, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations*, 2014.

[5] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep learning models in natural language processing: A survey," 2019.

[6] W. Wang, L. Wang, R. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," 2020.

[7] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," 2016.

[8] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul 2018.

[9] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Melbourne, Australia), pp. 31–36, Association for Computational Linguistics, July 2018.

[10] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," 2018.

[11] Y.-T. Tsai, M.-C. Yang, and H.-Y. Chen, "Adversarial attack on sentiment classification," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Florence, Italy), pp. 233–240, Association for Computational Linguistics, Aug. 2019.

[12] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2018.

[13] H. Zhang, H. Zhou, N. Miao, and L. Li, "Generating fluent adversarial examples for natural languages," 2020.

[14] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," Internet Society, 2019.

[15] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," 2020.

[16] E. Wallace, P. Rodriguez, S. Feng, I. Yamada, and J. Boyd-Graber, "Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 387–401, Mar. 2019.

[17] Z. Zhou, H. Guan, M. Bhat, and J. Hsu, "Fake news detection via nlp is vulnerable to adversarial attacks," *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, 2019.

[18] D. Khashabi, T. Khot, and A. Sabharwal, "More bang for your buck: Natural perturbation for robust question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Nov. 2020.

[19] "Human adversarial qa: Did the model understand the paragraph?,"

[20] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," 2018.

[21] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, “Black-box generation of adversarial text sequences to evade deep learning classifiers,” 2018.

[22] S. Ren, Y. Deng, K. He, and W. Che, “Generating natural language adversarial examples through probability weighted word saliency,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 1085–1097, Association for Computational Linguistics, July 2019.

[23] J. Xu, L. Zhao, H. Yan, Q. Zeng, Y. Liang, and X. Sun, “LexicalAT: Lexical-based adversarial reinforcement training for robust sentiment classification,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), Nov. 2019.

[24] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” 2020.

[25] B. Alshemali and J. Kalita, “Generalization to mitigate synonym substitution attacks,” in *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, Nov. 2020.

[26] S. Tan, S. Joty, M.-Y. Kan, and R. Socher, “It’s morphin’ time! Combating linguistic discrimination with inflectional perturbations,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), July 2020.

[27] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” in *2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), 2017.

[28] Y. Wang and M. Bansal, “Robust machine comprehension models via adversarial training,” June 2018.

[29] N. J. Nizar and A. Kobren, “Leveraging extracted model adversaries for improved black box attacks,” 2020.

[30] W. Wu, D. Arendt, and S. Volkova, “Evaluating neural machine comprehension model robustness to noisy inputs and adversarial attacks,” 2020.

[31] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, “Adversarial example generation with syntactically controlled paraphrase networks,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), June 2018.

[32] C. Si, Z. Yang, Y. Cui, W. Ma, T. Liu, and S. Wang, “Benchmarking robustness of machine reading comprehension models,” 2020.

[33] W. C. Gan and H. T. Ng, “Improving the robustness of question answering systems to question paraphrasing,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 6065–6075, Association for Computational Linguistics, July 2019.

[34] Y. Zhang, J. Baldridge, and L. He, “PAWS: Paraphrase adversaries from word scrambling,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), June 2019.

[35] S. Garg and G. Ramakrishnan, “BAE: BERT-based adversarial examples for text classification,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 6174–6181, Association for Computational Linguistics, Nov. 2020.

[36] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, “BERT-ATTACK: Adversarial attack against BERT using BERT,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 6193–6202, Association for Computational Linguistics, Nov. 2020.

[37] V. Malik, A. Bhat, and A. Modi, “Adv-olm: Generating textual adversaries via olm,” 01 2021.

[38] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, “Generating natural language adversarial examples,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 2890–2896, Association for Computational Linguistics, Oct.-Nov. 2018.

[39] Z. Meng and R. Wattenhofer, “A geometry-inspired attack for generating natural language adversarial examples,” 2020.

[40] T. Le, S. Wang, and D. Lee, “Malcom: Generating malicious comments to attack neural fake news detection models,” *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 282–291, 2020.

[41] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, “TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Oct. 2020.

[42] J. Y. Yoo, J. Morris, E. Lifland, and Y. Qi, “Searching for a search method: Benchmarking search algorithms for generating NLP adversarial examples,” in *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Nov. 2020.

[43] B. Edizel, A. Piktus, P. Bojanowski, R. Ferreira, E. Grave, and F. Silvestri, “Misspelling oblivious word embeddings,” 2019.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.

[45] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2020.

[46] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” 2020.

[47] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019.

[48] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, “TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 119–126, Association for Computational Linguistics, Oct. 2020.

[49] S. Garg and G. Ramakrishnan, “Bae: Bert-based adversarial examples for text classification,” 2020.

[50] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, “Bert-attack: Adversarial attack against bert using bert,” 2020.

[51] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, “Certified robustness to adversarial word substitutions,” 2019.

[52] D. Pruthi, B. Dhingra, and Z. C. Lipton, “Combating adversarial misspellings with robust word recognition,” 2019.

[53] J. Li, S. Ji, T. Du, B. Li, and T. Wang, “Textbugger: Generating adversarial text against real-world applications,” *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.