BinarySelect to Improve Accessibility of Black-Box Attack Research

Shatarupa Ghosh and **Jonathan Rusert***
Purdue University, Fort Wayne
shatarupa.ghosh012@gmail.com, jrusert@pfw.edu

Abstract

Adversarial text attack research is useful for testing the robustness of NLP models, however, the rise of transformers has greatly increased the time required to test attacks. Especially when researchers do not have access to adequate resources (e.g. GPUs). This can hinder attack research, as modifying one example for an attack can require hundreds of queries to a model, especially for black-box attacks. Often these attacks remove one token at a time to find the ideal one to change, requiring n queries (the length of the text) right away. We propose a more efficient selection method called *BinarySelect* which combines binary search and attack selection methods to greatly reduce the number of queries needed to find a token. We find that *BinarySelect* only needs $\log_2(n) * 2$ queries to find the first token compared to n queries. We also test *BinarySelect* in an attack setting against 5 classifiers across 3 datasets and find a viable tradeoff between number of queries saved and attack effectiveness. For example, on the Yelp dataset, the number of queries is reduced by 32% (72 less) with a drop in attack effectiveness of only 5 points. We believe that *BinarySelect* can help future researchers study adversarial attacks and black-box problems more efficiently and opens the door for researchers with access to less resources.

1 Introduction

Adversarial text attacks have seen a surge in research in recent years (Qiu et al., 2022). Attacks help to test the robustness of Natural Language Processing (NLP) models by both testing words and syntactic structures an NLP model might not be familiar with (Iyyer et al., 2018; Qi et al., 2021), as well as simulating attacks that humans may use to trick the NLP model (Formento et al., 2023; Wang et al., 2022).

Adversarial attacks assume some level of knowledge of the models they target. White-box attacks (Sadrizadeh et al., 2022; Choi et al., 2021) have access to a model's weights and architecture. This allows the attack to more quickly find the tokens which the classifier is leveraging, however, this may be unrealistic when considering models deployed online. Black-box attacks (Deng et al., 2022; Le et al., 2022) have access only to the output (e.g. label) and probabilities (or logits) of a model. This restriction means that black-box attacks spend more time querying the model to find the same tokens.

In the case of text classification, attacks often remove or mask one token (word) at a time and check the change in probability (Jin et al., 2020; Li et al., 2020; Ren et al., 2019; Formento et al., 2023). In a text of length n, this results in n number of queries before the attack even starts to change the text. For longer texts, this can slow down attacks. For researchers with access to fewer resources (e.g. no or few GPUs), this can greatly hinder verifying attacks, or other related research (e.g. attack defense or attack detection). In this research we propose a new method, *BinarySelect*, to reduce the number of queries required to find the most relevant words[1] to the model.

BinarySelect is inspired by the binary search algorithm. Whereas binary search requires a sorted list of values, *BinarySelect* uses the probabilities returned from the classifier to guide its search. Specifically, *BinarySelect* removes the first half of the text and compares the change in the probability to removing the second half. The half that causes the larger drop becomes the new search area and the algorithm repeats until 1 word (or token) remains. This algorithm greatly speeds up finding the first relevant word. Furthermore, to reduce future queries the algorithm leverages a binary tree to hold probabilities that have already been found. We find a tradeoff with *BinarySelect* with a small reduction in effectiveness but a large reduction in number of queries. Though we focus on adversarial

*Corresponding Author

[1]We focus on words, and verify on characters later on.

Proceedings of the 31st International Conference on Computational Linguistics, pages 10960–10976
January 19–24, 2025.

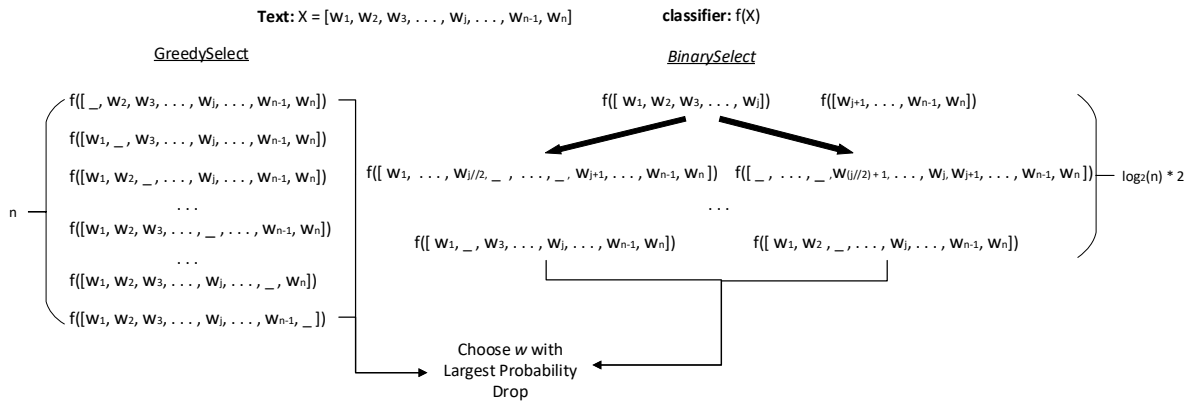

Figure 1: Visualization of GreedySelect versus *BinarySelect*. GreedySelect removes 1 word at a time and checks the change in probability. *BinarySelect* continuously splits the text in 2 and excludes the segments from the query. The excluded segment which causes the highest drop in target class probability is split again and so on. Eventually, the splitting leaves only 1 word which is chosen.

attacks, *BinarySelect* could also be leveraged for other black-box NLP models.

Our research makes the following contributions:

1. Propose a new selection algorithm, *BinarySelect*, to reduce the number of queries required by adversarial attacks and make attack and related research more accessible to others.

2. Explore and verify the theoretical effectiveness of *BinarySelect* in finding the most relevant words in text classification. We find that *BinarySelect* is able to find the first relevant word in $\log_2(n) * 2$ queries, which strongly outperforms the previous GreedySelect at n queries.

3. Evaluate *BinarySelect* as a tool in adversarial attacks for 3 text classification datasets against 5 text classification systems. We find that *BinarySelect* offers a strong tradeoff by reducing the average number of queries by up to 60% with a smaller drop in attack effectiveness.

Overall *BinarySelect* provides an alternative selection method for black-box algorithms. It provides an easy way to balance number of queries with attack effectiveness to allow researchers with lower resources a place in the field[2].

2 Proposed Approach

In this section we define our proposed selection method, *BinarySelect*. For background, we first define the goal of a word selection method and then define the approach commonly used by previous black-box attack research[3], which we call *GreedySelect*. A visualization of the difference between the two methods can be found in Figure 1. Note that we test our proposed method in the area of text classification, so terminology focuses on this area specifically moving forward.

2.1 Threat Model

The approaches assume black-box knowledge of a model. Specifically, no knowledge of model architecture or weights are known. Approaches are able to send queries to the model and the model returns a label (if classification) and confidence score. These assumptions follow previous black-box adversarial attack research in NLP (Alzantot et al., 2018; Garg and Ramakrishnan, 2020; Li et al., 2020; Gao et al., 2018; Hsieh et al., 2019; Li et al., 2021). Note some prior research has referred to this as "grey-box" due to probability access.

2.2 Selection Methods

Let a text of length n, be represented as $X = \{w_1, w_2, ..., w_n\}$, where w_i is the i-th word in the text. The goal of a selection method is to return the word w_j (or token) which has the greatest impact on a classifier's decision (or probability). Note that w_j is generally then replaced with a new word or modified to hurt the classifier in its ability to make the best decision. After replacement, the selection method then returns the word with the second-highest impact on the classifier's decision, and so on.

[2]Our code can be found at https://github.com/JonRusert/BinarySelect

[3]More related work found in Appendix B

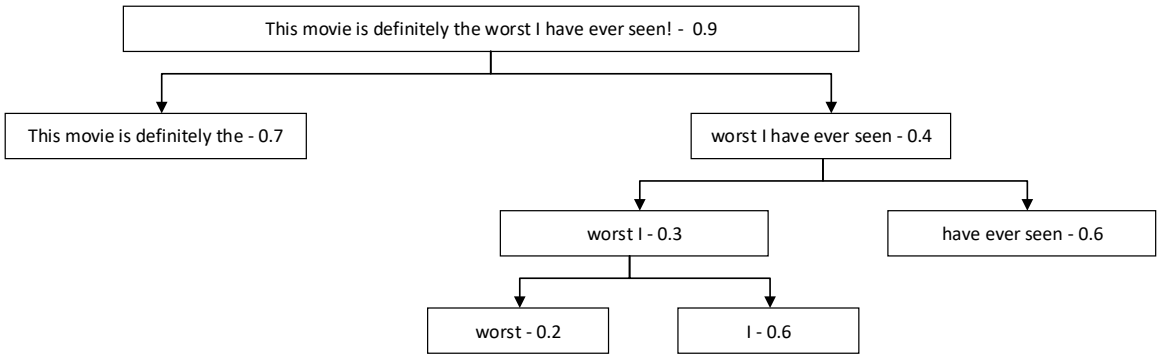

Figure 2: Visualization of Binary Tree leveraged to store the probabilities returned from *BinarySelect*. Note that the values indicate probability of target class. After the root node, the probabilities are calculated by removing the text at that node from the original text.

2.3 Greedy Select

Greedy select and variations have been strongly leveraged by previous black-box attacks (Ren et al., 2019; Li et al., 2020; Hsieh et al., 2019; Formento et al., 2023; Garg and Ramakrishnan, 2020; Gao et al., 2018; Jin et al., 2020; Li et al., 2019; Rusert et al., 2022). This method deletes one word at a time (with replacement) and checks the change in classifier probability. More formally, let f represent the classifier, and $f(X)$ represent the classification score (i.e. probability). For each word w_i in X, greedy select removes the word and finds Δ_i:

$$\Delta_i = f(X) - f(X//w_i) \quad (1)$$

which is the change in probability of a target class when w_i is removed. The word with the highest drop in target class probability is selected as the word to replace. Note here we define it with deletion, however, some of the attacks replace the word with masks instead (Li et al., 2020). Furthermore, other variations include scores of different portions of the text (Gao et al., 2018). Nevertheless, all the related methods remove one word at a time to find the greatest drop. This results in at least n queries.

2.4 *BinarySelect*

BinarySelect (Algorithm in Appendix A) adopts a more systematic and targeted approach. By dividing the sentence and evaluating classification score changes, it greatly narrows down the search space. This technique builds off of the binary search algorithm by viewing the texts as larger segments to search.

In *BinarySelect*, the text is continuously partitioned into two segments until a segment containing a single word is reached. At each step, the method evaluates the impact of excluding each segment on the classifier's output probability and selects the segment that results in the greatest drop in probability. More formally, let f represent the classifier, and $f(X)$ represent the classification score (i.e., probability). Given an input text X, *BinarySelect* partitions X into two segments, X_1 and X_2, such that $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$. The method calculates the probabilities $f(X_1)$ and $f(X_2)$ by excluding X_2 and X_1, respectively, from the original text X. The difference in probability for each segment is computed as:

$$\Delta_i = f(X) - f(X_i) \quad (2)$$

where Δ_i represents the change in the probability of a target class when segment X_i is excluded from the input text X. The segment with the highest probability drop of the target class, is processed further. If the segment is a single word, then this word is chosen as the most influential word. If it is not, then the process repeats with the segment becoming the next text to be divided in two.

2.5 *BinarySelect* - Retaining Memory

For GreedySelect, repeating the word selection stage is simple since all probability drops are calculated in the first pass through. However, *BinarySelect* only has a score for a single word in the text. To avoid additional queries, we leverage a binary tree structure to keep track of which segments the algorithm has generated scores for.

A visualization of this can be seen in Figure 2. This structure is continually updated as new segments are queried against the classifier (described

in Section 2.4). During the selection step, *BinarySelect* explores the tree path with the greatest drop in probability which hasn't been fully explored.

3 Theoretical Performance

We examine the theoretical performance of *BinarySelect* by examining 3 cases:

Best Case: In the best case, only one word is needed to be found. In this case, *BinarySelect* takes at most $\log_2(n) * 2$ queries. This is because it takes $\log_2(n)$ splits to reach a single word and each split requires 2 queries to guide the method. This value is less than GreedySelect which takes n queries for even 1 word, since it needs to remove every word and test the probability changed.

Average Case: Since each dataset and classifier can rely on a different number of words for classification, it can be difficult to know how many words are relied upon for classification on average. We estimate this value based on previous research. BERT-Attack (Li et al., 2020), reports the percentage of a text that is perturbed during the attack for IMDB and AG News (Section 6). This percentage is 4.4 for IMDB and 15.4 for AG News. We use this to estimate the number of words changed on average (percentage X average text length). For IMDB this results in an average of 9.5 words being changed (average text length of 215) and 6.6 words for AG News (average text length of 43). This means to find the words, GreedySelect would need the average number of words as queries (215 and 43).

For *BinarySelect* it is non-deterministic, since any query after the first, will leverage the binary structure (Appendix 2.5). The first query is again $\log_2(n) * 2$ queries. For the second query the first split and query is not needed as it was estimated previously. In the worst case scenario, the half of the tree that wasn't explored contains the next largest drop in probability and thus it is expanded. This means, in the worst case, the second query requires $\log_2(n/2) * 2$ queries. We can follow this worst case scenario as a basis to estimate the number of queries needed for *BinarySelect*. This results in a value of $\log_2(n) * 2 + \log_2(n/2) * 2 + \log_2(n/4) * 2 + ... + \log_2(n/(2^k - 1)) * 2$. Note that when $2^k - 1$ is larger than n, then we cap the value at 1, since at the lowest level, we need 2 queries for the two words being split.

For IMDB, with an average of 10 (9.6) word changes this results in 72 queries. For AG News, with an average of 7 (6.6) word changes, this value is 37. Note that this value would only be lower if the next most probable words are in the already explored structure. We can then see that even in the average (worst-case) *BinarySelect* requires less queries to find the same number of words as *GreedySelect*. We can also note a stronger performance increase in longer texts.

Worst Case: In the extreme worst case, we need to find the probabilities of every word in the input text. For GreedySelect, this is again equal to n. However, for *BinarySelect*, this is much greater as it will make n queries, as well as each split level queries. This results in $n + \sum_{i=1}^{\log_2(n)} n/(2^i)$. This scenario shows a disadvantage of *BinarySelect* and shows that it is not a permanent fix for *GreedySelect*, especially in scenarios where replacements are needed for a large percentage of the text.

Token #	AG News	IMDB
1	12.5	17.2
2	17.9	25.0
3	21.6	30.9
4	24.4	35.8
5	26.7	40.0
6	29.0	44.0
7	30.9	47.6
8	32.7	51.0
9	34.5	54.2
10	35.9	57.0
GS	39.5	230.6

Table 1: Average Queries to find the k top words for BS. Since GS requires all words to be checked, the number of queries is the same for all 10.

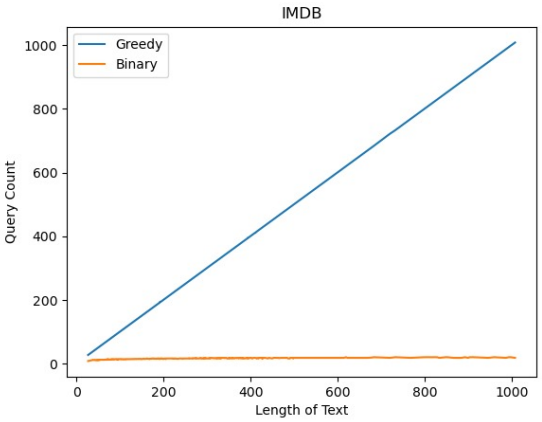

Figure 3: Number of queries required to find a word leveraged by the classifier. GreedySelect's queries increase linearly, while *BinarySelect* shows a log trend.

4 Validation of BinarySelect

4.1 Verification of Theoretical Performance

We validate the theoretical performance by running the binary select algorithm on 1000 AG News and IMDB (Section 6) examples, using a fine-tuned ALBERT (Devlin et al., 2019) classifier for feedback. We retrieve the top 10 words and note how many queries were required for each word. The results can be seen in Table 1. We find that on average the number of queries is less than the estimated amount in the theoretical case. For AG News, the estimated average value was 37, while the found average was 30.9 (7 words). Similarily, for IMDB, the estimated value was 72 and the found value was 57. This is because in the estimated value, we looked at the worst case of the average case where an unexplored branch of the binary structure was explored every query. However, this was not the case in the experiments. In both cases, the needed queries is lower than what is needed for the GreedySelect for all 10 queries.

We further validate *BinarySelect*'s advantage by examining how many queries the method needs to find the most influential word. We do this for the 1000 IMDB examples and note the length of the text and number of queries required to find the most influential word. These results can be found in Figure 3. We can observe that the GreedySelect has a linear trend, while the *BinarySelect* follows a log trend, this indicates a significant difference in computational complexity between the two methods. Specifically, it suggests that as the length of the input sentence increases, the query count for the GreedySelect tends to increase linearly. In contrast, the *BinarySelect* demonstrates a more consistent log query count regardless of sentence length. This observation emphasizes the efficiency advantage of *BinarySelect* over GreedySelect, particularly when dealing with longer input texts. The log trend for *BinarySelect* suggests that its computational requirements remain relatively stable and independent of input size, which can be a highly advantageous characteristic in practical applications.

4.2 Agreement between BinarySelect and GreedySelect

It is not sufficient for BinarySelect to find influential words more efficiently than GreedySelect, we also need BinarySelect to find words that are truly relevant. To verify this, we run 2 more experiments which measure on if GreedySelect and BinarySelect agree on the most influential words.

		BinarySelect	Random
	Position of 1st GS Token		
AG	Average	2.3	5.0
	Median	1	5
	Not Found	255	647
IMDB	Average	2.9	5.1
	Median	2	4
	Not Found	583	921
	Num. Overlaps with GS top 10		
AG	Average	5.7	3.4
	Median	6	3
	None	4	15
IMDB	Average	3.7	1.7
	Median	4	1
	None	110	345

Table 2: Comparison between the top 10 words found by GreedySelect (GS). Position of 1st refers to which position the top GS token is found by the respective method (low values desired). Num. Overlaps refers to the number the top 10 GS tokens appear in the top 10 found by the other method (higher values desired).

First, we take the top word given by GS and note which influential position it was given by BS. If GS and BS always agree on the most influential word, then that position will be 1. We examine the top 10 words found by BS for both the AG News and IMDB examples. We also make a random baseline which randomly choose 10 words in the input texts. The results can be found in Table 2. We find that for AG News, the most influential word found by GS appears in the 2.3 position on average and the 1 position for a median value. These are much lower than the random baseline which the position is 5 on average and median. In 255 of the texts, the top word of GS is not found in the top 10 BS list. This is also much lower than the random baseline which 647 texts do not include the word. For IMDB, these values are slightly larger (since the lengths of IMDB texts are much longer), with averages of 2.9 and 5.1 for BS and random respectively and median values of 2 and 4 respectively.

Second, we look at how many words in the top 10, BS and GS agree on. For the same examples, we note how many of the BS words occur in the GS list. For AG News, we find that the BS list has 5.7 of the same words on average (median of 6), which is more than the random at an average of 3.4 (median of 3). For IMDB these values are slightly lower (again due to IMDB text's lengths), on average the BS list contains 3.7 words (median of 4) which is still higher than the random baseline average of 1.7 (median of 1).

We see that though BS and GS do not completely agree on the most influential words, there is ample overlap between the two methods. Note that these experiments can only measure agreement and not which is the most "effective" in downstream tasks. To verify this, we leverage both BS and GS in a common setting, Adversarial Attacks (Section 5).

5 Testing *BinarySelect* in Adversarial Attacks

As noted, GreedySelect is widely leveraged by many black-box attacks (Ren et al., 2019; Li et al., 2020; Hsieh et al., 2019; Formento et al., 2023; Garg and Ramakrishnan, 2020; Gao et al., 2018; Jin et al., 2020; Li et al., 2019; Rusert et al., 2022). However, its need to examine every token in a text can slow down the attack algorithm greatly, which causes barriers for researchers with low resources. Thus, *BinarySelect* may be a strong replacement to decrease the overall number of queries required per attack.

As a reminder, in adversarial attacks, the goal is to create input examples that are understood (by humans) similarly to the original ones but lead to incorrect classifier predictions. This is often accomplished by modifying one word of a text at a time and checking against the classifier noting changes in probabilities. Once the modified text causes the classifier to fail, the attack ends. Note that the attacker must also aim to maintain the semantic integrity of the text to keep meaning.

5.1 Attack Description

To test the feasibility of *BinarySelect* in attacks, we create a similar attack framework to previous research. Our attack consists of two steps, word selection and word replacement:

1. Word Selection - the position of the word which the classifier relies on the most for classification is chosen to be replaced. Either *BinarySelect* or GreedySelect is used to find this position.

2. Word Replacement - the word at the selected position is replaced. We query WordNet for the selected word's synonyms. Each synonym is tested and the synonym which causes the classifier to fail or causes the greatest drop in target class probability is chosen. If the classifier does not fail with this replacement, the process repeats with Word Selection. However, this time the previous modified position is excluded as a candidate.

Note that this word replacement step is similar to PWWS (Ren et al., 2019). A more advanced replacement step would generate a stronger attack, however, we choose this simple replacement step to place the focus on the selection algorithms.

5.2 Restriction to k Words

To further improve the efficiency of the attack, we add the option of restricting the attack to modify at most k words. As k increases the attack effectiveness will naturally increase but the number of queries required will increase as well. Additionally, as more words change, the semantic integrity starts to weaken. This k is another useful tool for allowing researchers with lower resources to control effectiveness versus efficiency. We explore the effect of k in Section 8.

6 Experimental Setup

To evaluate *BinarySelect* in the adversarial attack setting, we run the attack (Section 5) on 5 classifiers across 3 datasets[4]. For space, the datasets and classifiers are described in detail in Appendix D.

6.1 Metrics

We use the following metrics to evaluate *BinarySelect* in the attack:

1. Accuracy - We measure the accuracy of each model before and after the attack for both GreedySelect (GS) and *BinarySelect* (BS). This helps measure the strength of the attack for each.

2. Average Queries - To measure the queries saved by using *BinarySelect*, we measure the number of queries needed for an attack on average. These queries indicate how many calls to the classifier are needed.

3. Average Queries when Successful - Similar to average queries, but in the cases when the attack is successful. *BinarySelect* will naturally suffer when more of the text is explored, which is what happens in failed attacks. This measurement shows an ideal case for the attack.

4. Effectiveness Differential Ratio (EDR) - To measure the tradeoff between Attack Success Rate (Equation 3)(ASR) and Average Queries, we propose EDR (Equation 6), which contrasts the percentage change in ASR (Equation 5) with the percentage change in Average Queries (Equation 4). We use this measure to help explore how k affects BS versus GS (Section 8).

[4]The majority of attacks are run on Google Colab and Kaggle which use NVidia K80 GPUs. Each attack combination took roughly 40 minutes.

		Albert		Distilbert		BERT		Roberta		LSTM	
		GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Yelp	Original Acc.	99.8		95.2		99.5		98.3		94.7	
	Attack Acc.	43.5	51.7	31.1	46.6	47.2	52.6	54.5	65.3	10.9	32.2
	Avg. Queries	217	150	208	141	222	150	239	172	181	119
	Avg. Q's (Success)	156	93	162	93	150	100	160	107	173	91
IMDB	Original Acc.	97.7		96.8		97.9		97.6		84.8	
	Attack Acc.	51.8	66.9	37.2	58.2	54.4	70.0	55.0	72.5	25.4	52.9
	Avg. Queries	318	172	305	156	317	173	332	182	274	136
	Avg. Q's (Success)	273	106	265	99	269	110	275	113	262	96
AG News	Original Acc.	98.8		97.4		99.6		99.2		93.1	
	Attack Acc.	46.2	48.2	60.7	62.8	62.6	64.4	55.9	58.3	43.5	47.7
	Avg. Queries	111	111	121	124	125	127	119	121	104	112
	Avg. Q's (Success)	84	76	92	86	89	84	86	82	84	85

Table 3: Adversarial Attack Results when $k = 15$. "Original Acc." is the original accuracy of the model, "Attack Acc." is the model accuracy on the text modified by the attack. "Avg. Queries" is the average number of queries used, "Avg. Q's (Success)" are the number of queries used for successful attacks. GS - GreedySelect, BS - *BinarySelect*.

$$\text{ASR} = \frac{Original_{Acc.} - Attack_{Acc.}}{Original_{Acc.}} \quad (3)$$

$$\text{Query}_{\text{Diff}} = \frac{\text{Queries}_{\text{Greedy}} - \text{Queries}_{\text{Binary}}}{\text{Queries}_{\text{Greedy}}} \quad (4)$$

$$\text{ASR}_{\text{Diff}} = \frac{\text{ASR}_{\text{Binary}} - \text{ASR}_{\text{Greedy}}}{\text{ASR}_{\text{Greedy}}} \quad (5)$$

$$\text{EDR} = \text{ASR}_{\text{Diff}} + \text{Query}_{\text{Diff}} \quad (6)$$

7 Results

The main results for our attack experiments can be found in Table 3. For each classifier, we compare the GreedySelect (GS) and *BinarySelect* (BS). We use a k value[5] of 15, which means the attack was limited to replacing 15 words in a text at most (a further exploration of k values can be found in Section 8). The first three metrics described in 6.1 are shown. The following observations are made:

***BinarySelect* reduces the number of queries greatly, with some drop in attack effectiveness.** When examining Table 3, we see drops in query amounts for both IMDB and Yelp datasets. Focusing on the Albert classifier, we see a 31% difference in queries between GreedySelect (217) and *BinarySelect* (150). This drop in queries causes a slight drop in attack effectiveness of 16%. Similarly for IMDB, we see a larger difference in queries. Specifically, *BinarySelect* causes a 46% reduction in queries compared to GreedySelect, with a 23% drop in attack effectiveness. If we focus on the number of queries for successful attacks, then this increases to a 54% reduction in queries for the same 23% effectiveness tradeoff. Hence, we see a stronger positive effect in query reduction compared to attack effectiveness.

***BinarySelect* is less effective on shorter texts.** For AG New, we see similar results between GreedySelect and *BinarySelect*. Both the query numbers and accuracy are within a few points of each other. A main reason is that AG News contains shorter texts on average (43 words) compared to Yelp (157) and IMDB (215). This means *BinarySelect* will save less queries with each search for AG News compared to Yelp and IMDB. Nonetheless, we see that in the extended case, *BinarySelect* still achieves comparable performance to GreedySelect.

***BinarySelect* strongest effect is demonstrated on Yelp Dataset.** The results on the Yelp dataset show a clear instance in the strength of *BinarySelect*. We see a 32% reduction in queries for BERT, a 31% reduction for Albert, and a 32% reduction for Distilbert. The effectiveness of the attack is at a lower rate as well, for example a 10% drop for BERT and 16% for albert. Distilbert seems to be an exception with a larger drop in effectiveness. These results demonstrate the true potential for *BinarySelect*.

Successful attacks cause a greater reduction in attack queries. When examining the average queries for the succesful attacks, we see a greater reduction in query amount on average. For IMDB, the reductions increase from 46% (Albert), 45% (XLNet), and 45% (RoBERTa) to 61% (Albert), 60% (XLNet), and 59% (RoBERTa). This indicates that if a more successful replacement step is

[5]Tables for other k can be found in Appendix F

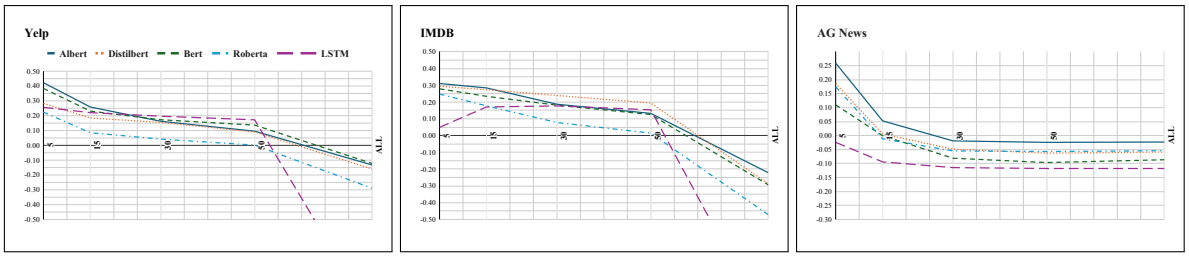

Figure 4: Effect of k Values on EDR (Equation 6) for the successful attacks. Positive values indicate a better trade-off between reduction in queries versus loss of accuracy drop for BS.

	$k=5$		$k=15$		$k=30$		$k=50$		$k=ALL$	
	GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Orig Acc.	85.8									
Attack Acc.	47.8	56.0	29.2	38.5	24.1	32.0	22.8	28.9	22.9	27.9
Avg. Queries	108	31	112	50	117	68	122	85	135	133
Avg. Q's (Success)	101	23	108	34	111	42	113	47	113	49

Table 4: Character Level attack using BS and GS against canine-s finetuned on SST2 data.

chosen, then the algorithm will increase in effectiveness.

In all observations we see a clear reduction in query amounts with a lesser reduction in attack effectiveness. This helps highlight the tradeoffs of *BinarySelect*. Future work would implement the algorithm with more effective replacement/modification steps to extend *BinarySelect* to its full potential.

8 Choosing an Effective k

The main attack results use $k = 15$, however, the chosen k will impact both attack effectiveness and queries needed. Different datasets will benefit from different k. To investigate this, for each dataset and classifier, we test $k = \{5, 15, 30, 50, \text{ALL}\}$ where "ALL" imposes no restrictions on the number of words to replace. Figure 4 shows the effect of different k for successful attacks, measured with EDR (Equation 6). As k increases the query amount increases and the accuracy of the targeted classifier decreases. This causes a better trade off of EDR at lower k. The optimal k will minimize the accuracy and minimize the amount of queries We see that $k = 15, 30$ offers a balance for Yelp and IMDB, but $k = 5$ would be better for AG News. These results demonstrate the need for testing different k for different tradeoff goals.

9 Verification: Character-level Attack

We verify the main results of *BinarySelect* by extending the experiments to character-level attacks. Specifically, we target a character-level model (Clark et al., 2021), fine-tuned on the SST2 (Stanford Sentiment Treebank, contains movie reviews labeled for sentiments) dataset. We leverage BS and GS to choose which character to modify, and use the ECES unicode replacement from VIPER (Eger et al., 2019) to replace each chosen character. We run the attack for $k = 5$ and $k = ALL$ on the validation set (872 instances). The results can be found in Table 4.

We observe results consistent (or better) with the word level attacks (Section 7). For the lower k, we see *BinarySelect* outperform GreedySelect, while for large k, we see similar attack effectiveness and similar query counts. However, again for the successful cases, BS strongly outperforms GS, which further points to the potential strengths of BS.

10 Further Analysis: Combining *BinarySelect* and GreedySelect

We see that in our results (Section 7), there exists a tradeoff between using *BinarySelect* and GreedySelect. Furthermore, the chosen k greatly affects this tradeoff. Specfically, we see a better EDR with lower k. This means, that *BinarySelect* is a better choice for texts which require less word changes. To further examine this, we imagine an **oracle** model which knows how many words need to be changed for an attack to succeed (We compare the modified texts by *BinarySelect* to determine the number). The oracle can leverage the strengths of both *BinarySelect* and GreedySelect. If the number of words to be changed is less than j, then *BinarySelect* is used, otherwise GreedySelect is used.

Model	Attack Acc.	Avg. Q's
GS	3.4	407
BS	3.8	526
Oracle		
j <= 5	3.4	369
j <= 15	3.4	346
j <= 30	3.4	341
j <= 50	3.4	358
j > 50	3.8	575

Table 5: Combination results for GS and BS on IMDB data for DistilBert. Oracle knows how many words need to be perturbed for an attack and uses BS for texts less than j and GS for those more than j.

Model	Attack Acc.	Avg. Q's
GS	3.4	407
BS	3.8	526
Oracle		
j <= 5	3.4	369
j <= 15	3.4	346
j <= 30	3.4	341
j <= 50	3.4	358
j > 50	3.8	575
Confidence Model		
Score <= Avg_5	3.4	422
Score <= $\text{Avg}_1 5$	3.4	419
Score <= $\text{Avg}_3 0$	3.4	422
Score <= $\text{Avg}_5 0$	3.4	422
Score > $\text{Avg}_5 0$	3.4	511

Table 6: Combination results for GS and BS on IMDB data for DistilBert. Oracle knows how many words need to be perturbed for an attack and uses BS for texts less than j and GS for those more than j.

We compare this **oracle** model with the previous results (k = ALL) for DistilBert on IMDB with various j in Table 5.

As can be observed, the oracle is able to achieve GS's lower accuracy, with much lower queries overall by leveraging both BS and GS effectively. This **oracle** model is the ideal to strive for, however, we do not automatically know how many words will need to be changed for a target model to fail. We run a preliminary experiment to determine if confidence score can be utilized as an oracle (Appendix 10.1), but find it does not perform as well as the oracle. Future work will further investigate discovering this automatic oracle to most effectively utilize BS and GS.

10.1 Confidence Model to Emulate Oracle

As a preliminary, we look at the confidence (probabilities) of the classifier on the original text as an indicator. When binning the average confidence scores against the number of word changes, we find a slight pattern of increase: [5 - 93.77, 15 - 96.27, 30 - 97.12, 50 - 97.46, ALL - 97.67]. However, in the larger changes, there exists slight differences in the average confidence scores. Nonetheless, we try a secondary model which uses the noted confidence scores to determine if BS or GS is used. Similarly to **Oracle**, if the original confidence score is less than the average confidence for a bin, then BS is used, otherwise GS. Note that this is still part oracle, as the average confidence scores would not be known. These results are found in Table 6. As can be observed, the confidence model performs better than BS but similarly to GS, and not as well as the **Oracle** model. This means the confidence score alone is not adequate to determine when to use BS versus GS.

11 Conclusion

BinarySelect shows a strong promise to increase efficiency of attack research and other related domains. Specifically, we found that *BinarySelect* is able to find a word relevant to a classifier in $\log_2(n) * 2$ steps. This is much more efficient that GreedySelect and its variants which take n (or more) queries to produce a word.

We further tested *BinarySelect* in the downstream task of adversarial attacks. To keep focus on the selection method, we combined it with a WordNet replacement method. We found a viable tradeoff between query reduction and drop in attack effectiveness. For BERT on the Yelp dataset, *BinarySelect* takes 32% (72) less queries than GreedySelect with only a 10% (5 point) drop in attack effectiveness. Furthermore by including the choice for a k, we introduced more control to the researcher. We further verified this on a character-level attack. Finally, we showed the potential for ideal method that combines *BinarySelect* and GreedySelect, however, it is left to future research to fully solve this problem.

GreedySelect's frequent usage in multiple eminent attacks is resource draining. *BinarySelect* is effective in giving low-resource researchers the ability to be apart of this domain, allowing the best ideas a chance to be realized.

12 Limitations

Here we note limitations of our study for future researchers and users to consider:

1. **Stronger Replacement Steps Exist for Attacks** - Our algorithm was limited in measurement due to leveraging WordNet alone as replacement.

Other attack research has leveraged transformer models such as BERT to give more relevant suggestions for replacement. This could have resulted in earlier stopping in the attack due to better replacement choices. However, since the main focus was on the selection method, we purposely chose a simple replacement method to showcase it. Indeed, future researchers will apply *BinarySelect* with their replacement algorithms for stronger attack research.

2. **Human Validation of Choices** - In the pilot study, we compare *BinarySelect* to GreedySelect. While *BinarySelect* clearly exhibits a stronger performance in terms of queries, it is not known to what extent the top or (top X) word is retrieved. Part of this issues lies with the goal of the selection methods. Since selection methods are basing their decision off of classifier feedback rather than human feedback, we cannot simply ask humans which words are most beneficial to the classification. This is because classifiers do not always choose the terms humans consider. This explainabilty of models is an open problem in and of itself. Still incorporation of *BinarySelect* into other downstream related tasks could help verify its selection strength.

13 Ethical Considerations

One must always take into consideration the negative uses of research. This is especially relevant when dealing with adversarial attacks. A malicious user may take *BinarySelect* and use it to improve a system which targets or harasses others. However, we believe the positive uses of our proposed algorithm outweigh the negative uses. This is especially true since this algorithm and code is made available to the public, which allows other researchers to build on or research defenses against it. Furthermore, the second stage of the attack is simple and therefore already known in this space. We believe these reasons along with the potential positives of allowing researchers with low access to computational power, to justify publishing.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. OutFlip: Generating examples for unknown intent detection with natural language attack. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 504–512, Online. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. CANINE: pre-training an efficient tokenization-free encoder for language representation. *CoRR*, abs/2103.06874.

Chuyun Deng, Mingxuan Liu, Yue Qin, Jia Zhang, Hai-Xin Duan, and Donghong Sun. 2022. ValCAT: Variable-length contextualized adversarial transformations using encoder-decoder language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1735–1746, Seattle, United States. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. 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)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Brian Formento, Chuan Sheng Foo, Luu Anh Tuan, and See Kiong Ng. 2023. Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1–34, Dubrovnik, Croatia. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate speech detection. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529, Florence, Italy. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. 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)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2953–2965, Dublin, Ireland. Association for Computational Linguistics.

Yibin Lei, Yu Cao, Dianqi Li, Tianyi Zhou, Meng Fang, and Mykola Pechenizkiy. 2022. Phrase-level textual adversarial attack with label preservation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1095–1112, Seattle, United States. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, Online. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. *Proceedings 2019 Network and Distributed System Security Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. 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*, pages 119–126, Online. Association for Computational Linguistics.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shilin Qiu, Qihe Liu, Shijie Zhou, and Wen Huang. 2022. Adversarial attack and defense technologies in natural language processing: A survey. *Neurocomputing*, 492:278–307.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Jonathan Rusert, Zubair Shafiq, and Padmini Srinivasan. 2022. On the robustness of offensive language classifiers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7424–7438, Dublin, Ireland. Association for Computational Linguistics.

Sahar Sadrizadeh, Ljiljana Dolamic, and Pascal Frossard. 2022. Block-sparse adversarial attack to fool transformer-based text classifiers. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7837–7841.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Boxin Wang, Chejian Xu, Xiangyu Liu, Yu Cheng, and Bo Li. 2022. SemAttack: Natural textual attacks via different semantic spaces. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 176–205, Seattle, United States. Association for Computational Linguistics.

Shangyu Xie and Yuan Hong. 2022. Differentially private instance encoding against privacy attacks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 172–180, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples.

A Binary Select

The full algorithm for BinarySelect can be found in Algorithm 1.

Algorithm 1 Binary Select

Require: $text$
Ensure: $most\_influential\_pos$
$Score_{Orig} \leftarrow Classifier(text)$
$start \leftarrow 0, end \leftarrow \text{len}(text) - 1$
while $start \neq end$ **do**
  $mid \leftarrow (start + end)//2$
  $left\_text \leftarrow text[0 : mid + 1]$
  $right\_text \leftarrow text[mid + 1 :]$
  $Score_{Left} \leftarrow Classifier(left\_text)$
  $Score_{Right} \leftarrow Classifier(right\_text)$
  $DropLeft \leftarrow Score_{Orig} - Score_{Left}$
  $DropRight \leftarrow Score_{Orig} - Score_{Right}$
  if $DropLeft > DropRight$ **then**
    $end \leftarrow mid$
  else
    $start \leftarrow mid + 1$
  end if
end while
$most\_influential\_pos \leftarrow start$

B Related Work

Adversarial text attacks are useful for testing robustness of models and even in areas of privacy concerns and censorship (Xie and Hong, 2022). Adversarial attacks are executed at different levels: 1. Character, 2. Word, 3. Phrase, 4. Sentence, 5. Multi-level.

Character-level attacks change individual characters in words to cause tokens to become unknown to the target NLP models. These attacks include addition/removal of whitespace (Gröndahl et al., 2018), replacement of visually similar characters (Eger et al., 2019), and shuffling of characters (Li et al., 2019). Word-level attacks replace words with synonyms that are less known to the target NLP models. The attacks have leveraged Word Embeddings (Hsieh et al., 2019), WordNet (Ren et al., 2019), and Mask Language Models (Li et al., 2020) to find relevant synonyms for replacement. Phrase-level attacks replace multiple consecutive words at once (Deng et al., 2022; Lei et al., 2022). Sentence-level attacks leverage generation methods to rewrite text in a format that the target NLP model is unfamiliar with (Ribeiro et al., 2018; Zhao et al., 2018). Multi-level attacks use a combination of the above attacks to cause model failure (Formento et al., 2023). We test our proposed methodology at the word-level, however, it could be extended to the character or phrase level easily.

Adversarial attacks have different levels of knowledge of their target model. White-box attacks are able to leverage complete model information, including the weights of the trained model and architecture (Sadrizadeh et al., 2022; Wang et al., 2022). Black-box attacks only have access to a models' confidence level (e.g. probabilities or logits) as well as their output (Le et al., 2022; Jin et al., 2020). In the case of text classification that output is the predicted label. Since white-box attacks have access to the weights of a model, they are able to find words to replace or modify very quickly. Black-box attacks however, need many queries since they only have access to classifier confidence which they check when making changes. As noted, our research aims to improve on previous black-box attacks by decreasing the number of queries needed to find the best words to replace.

C BS Structure Algorithm

Algorithm 2 shows the updated *BinarySelect* algorithm with use of the binary tree (BSNode) structure described in Section 5.

Algorithm 2 Binary Select

Require: text
Ensure: most_influential_pos
  Initialize BS structure with the root node representing the entire text and its corresponding classifier score.
  Initialize most_influential_pos to None.
  $ScoreOrig \leftarrow Classifier(text)$
  $DropMax \leftarrow 0$
  while BS structure is not fully explored **do**
    $cur\_node \leftarrow$ BS node with the lowest unexplored probability score
    if cur_node is a leaf node **then**
      Mark cur_node as explored
      if most_influential_pos is None or cur_node.prob < BS node at most_influential_pos.prob **then**
        for each word w in cur_node.data **do**
          $Scorew \leftarrow Classifier(text/w)$, where w is the word represented by cur_node
          $Dropw \leftarrow ScoreOrig - Scorew$
          if Dropw > DropMax **then**
            $DropMax \leftarrow Dropw$
            most_influential_pos $\leftarrow$ position of cur_node in the original text
          end if
        end for
      end if
    else
      Split cur_node's text segment into two parts
      Create left and right child nodes in the BS structure for the two parts
      Mark cur_node as explored
    end if
  end while
  return most_influential_pos

D Experimental Details: Datasets and Classifiers

To verify BinarySelect in an attack setting, we test it and GreedySelect against the following datasets and classifiers.

D.1 Datasets:

We test the attack on the following datasets, examined in previous attack research (Jin et al., 2020;

Li et al., 2020), randomly sampling 1000 examples from each test set:

1. Yelp Polarity - binary sentiment classification, containing texts from Yelp reviews. The labels are positive or negative. The average text lengths are 157 tokens.

2. IMDB - binary sentiment classification, containing text reviews for movies. Labels are positive or negative. The average text lengths are 215 tokens.

3. AG News - A multi-class (Sports, World, Business, Sci/Tech) dataset containing news texts. The average text lengths are 43 tokens.

D.2 Classifiers:

We test against 5 classifiers for each dataset, by leveraging pretrained TextAttack (Morris et al., 2020) and other Huggingface models[6]:

1. Albert (Lan et al., 2019) - a fine-tuned version of Albert, which shares weights across layers in order to obtain a smaller-memory footprint than BERT.

2. Distilbert (Sanh et al., 2020) - a fine-tuned Distilbert model. Distilbert was pretrained using BERT as a teacher for self-supervision and thus is a lighter, faster model than BERT.

3. BERT (Devlin et al., 2019) - a fine-tuned version of BERT-base-uncased. BERT pre-trains on next sentence prediction and masked language modelling tasks to gain an inherent understanding of text.

4. RoBERTa (Liu et al., 2019) - a fine-tuned version of RoBERTa. RoBERTa outperforms BERT in classification tasks, due to different choices in pretraining.

5. LSTM - LSTM trained on the respective datasets. The trained models are available from TextAttack[7].

E List of Huggingface Models

Table 7 contains the locations of the different models tested for our attack.

F k Results

We generate similar tables to Table 3 for $k = \{5, 15, 30, 50, \text{ALL}\}$. Table 8 is $k = 5$, Table 3 is $k = 15$, Table 9 is $k = 30$, Table 10 is $k = 50$, and Table 11 is $k = \text{ALL}$.

[6]Full list in Appendix E

[7]https://textattack.readthedocs.io/en/latest/3recipes/models.html

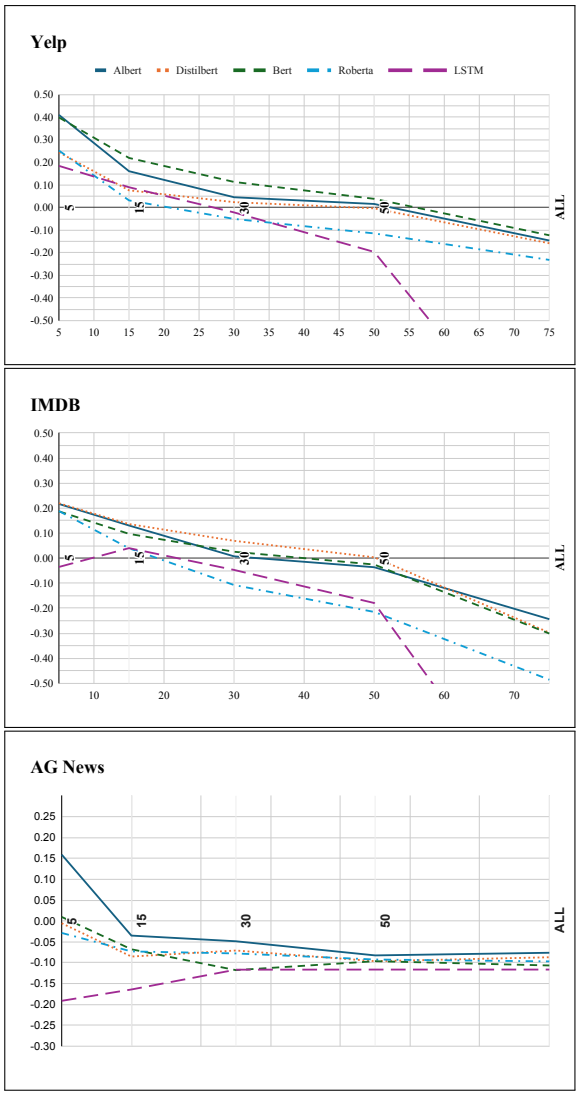

Figure 5: Effect of k Values on EDR (Equation 6) for the all attacks. Positive values indicate a better trade-off between reduction in queries versus loss of accuracy drop for BS.

G EDR Charts

Figure 5 shows the EDR values for different k values for both success and failed attacks. Trends are similar to Figure 4, although the failed attacks cause a lesser trade off for larger k.

	Model	Huggingface Location
Yelp	Albert	textattack/albert-base-v2-yelp
	Distilbert	randellcotta/distilbert-base-uncased-finetuned-yelp-polarity
	BERT	textattack/bert-base-uncased-yelp
	Roberta	VictorSanh/roberta-base-finetuned-yelp-polarity
	LSTM	lstm-yelp (TextAttack)
IMDB	Albert	textattack/albert-base-v2-imdb
	Distilbert	textattack/distilbert-base-uncased-imdb
	BERT	textattack/bert-base-uncased-imdb
	Roberta	textattack/roberta-base-imdb
	LSTM	lstm-imdb (TextAttack)
AG News	Albert	textattack/albert-base-v2-ag-news
	Distilbert	textattack/distilbert-base-uncased-ag-news
	BERT	textattack/bert-base-uncased-ag-news
	Roberta	textattack/roberta-base-ag-news
	LSTM	lstm-ag-news (TextAttack)

Table 7: The locations of pretrained models tested in our attack research.

		Albert		Distilbert		BERT		Roberta		LSTM	
		GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Yelp	Original Acc.	99.8		95.2		99.5		98.3		94.7	
	Attack Acc.	71.7	76.1	63.4	73.6	76.2	80.1	80.0	85.3	44.8	64.3
	Avg. Queries	170	74	172	75	171	74	178	81	166	70
	Avg. Q's (Success)	103	44	117	47	107	48	107	52	145	51
IMDB	Original Acc.	97.7		96.8		97.9		97.6		84.8	
	Attack Acc.	73.7	85.2	65.7	80.8	76.5	87.3	82.3	90.0	51.5	76.6
	Avg. Queries	267	81	266	78	265	82	273	84	254	71
	Avg. Q's (Success)	242	51	231	51	254	55	229	57	245	48
AG News	Original Acc.	98.8		97.4		99.6		99.2		93.1	
	Attack Acc.	76.6	77.6	85.5	87.6	85.7	88.1	81.9	85.1	76.7	82.1
	Avg. Queries	66	53	65	54	69	56	66	56	63	55
	Avg. Q's (Success)	53	37	53	33	53	38	53	34	54	37

Table 8: Adversarial Attack Results when $k = 5$. "Original Acc." is the original accuracy of the model, "Attack Acc." is the model accuracy on the text modified by the attack. "Avg. Queries" is the average number of queries used, "Avg. Q's (Success)" are the number of queries used for successful attacks. GS - GreedySelect, BS - *BinarySelect*.

		Albert		Distilbert		BERT		Roberta		LSTM	
		GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Yelp	Original Acc.	99.8		95.2		99.5		98.3		94.7	
	Attack Acc.	25.8	33.8.0	17.5	28.3	28.5	33.6	37	47.4	6.8	22.5
	Avg. Queries	261	220	233	195	270	220	298	263	184	155
	Avg. Q's (Success)	197	144	195	138	202	153	210	166	179	112
IMDB	Original Acc.	97.7		96.8		97.9		97.6		84.8	
	Attack Acc.	34.2	51.4	20.9	39.5	38.5	53.7	32.7	56.4	20.2	43.3
	Avg. Queries	369	266	337	231	373	268	384	285	284	196
	Avg. Q's (Success)	307	167	298	154	299	168	323	180	267	124
AG News	Original Acc.	98.8		97.4		99.6		99.2		93.1	
	Attack Acc.	23.9	23.9	33.0	32.1	32.9	34.7	28.8	29.2	21.9	22.1
	Avg. Queries	148	155	168	182	176	192	164	175	134	150
	Avg. Q's (Success)	117	119	135	143	139	146	128	135	112	125

Table 9: Adversarial Attack Results when $k = 30$. "Original Acc." is the original accuracy of the model, "Attack Acc." is the model accuracy on the text modified by the attack. "Avg. Queries" is the average number of queries used, "Avg. Q's (Success)" are the number of queries used for successful attacks. GS - GreedySelect, BS - *BinarySelect*.

		Albert		Distilbert		BERT		XLNet		Roberta	
		GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Yelp	Original Acc.	99.8		95.2		99.5		98.3		94.7	
	Attack Acc.	16.3	22.1	10.9	16.1	16.0	21.5	24.9	33.5	6.2	20.0
	Avg. Queries	295	270	248	233	307	275	348	347	186	194
	Avg. Q's (Success)	232	194	219	186	250	199	262	231	181	122
IMDB	Original Acc.	97.7		96.8		97.9		97.6		84.8	
	Attack Acc.	24.4	37.7	12.0	27.2	24.9	37.7	17.7	42.2	19.2	38.6
	Avg. Queries	417	356	359	294	425	361	422	383	293	259
	Avg. Q's (Success)	334	230	326	204	346	242	371	252	270	149
AG News	Original Acc.	98.8		97.4		99.6		99.2		93.1	
	Attack Acc.	17.2	18.3	23.3	23.7	23.5	24.2	18.5	20.0	16.2	16
	Avg. Queries	159	169	185	202	195	212	179	192	143	160
	Avg. Q's (Success)	132	134	155	164	159	173	152	158	124	139

Table 10: Adversarial Attack Results when $k = 50$. "Original Acc." is the original accuracy of the model, "Attack Acc." is the model accuracy on the text modified by the attack. "Avg. Queries" is the average number of queries used, "Avg. Q's (Success)" are the number of queries used for successful attacks. GS - GreedySelect, BS - *BinarySelect*.

		Albert		Distilbert		BERT		Roberta		LSTM	
		GS	BS	GS	BS	GS	BS	GS	BS	GS	BS
Yelp	Original Acc.	99.8		95.2		99.5		98.3		94.7	
	Attack Acc.	5.3	5.2	5.1	5.2	4.6	5.2	9.2	8.0	5.3	5.3
	Avg. Queries	372	427	271	313	372	415	476	592	196	421
	Avg. Q's (Success)	336	380	268	310	339	378	420	548	196	421
IMDB	Original Acc.	97.7		96.8		97.9		97.6		84.8	
	Attack Acc.	4.7	4.7	3.4	3.8	3.8	4.8	2.7	3.1	15.2	15.2
	Avg. Queries	571	709	407	526	578	746	489	724	353	756
	Avg. Q's (Success)	549	670	405	520	550	705	486	713	353	756
AG News	Original Acc.	98.8		97.4		99.6		99.2		93.1	
	Attack Acc.	17.1	18.0	23.2	23.2	23.1	24.2	18.3	20.4	16.2	16.0
	Avg. Queries	159	169	186	202	196	214	180	193	143	161
	Avg. Q's (Success)	133	134	156	164	161	172	153	157	124	139

Table 11: Adversarial Attack Results when $k =$ ALL. "Original Acc." is the original accuracy of the model, "Attack Acc." is the model accuracy on the text modified by the attack. "Avg. Queries" is the average number of queries used, "Avg. Q's (Success)" are the number of queries used for successful attacks. GS - GreedySelect, BS - *BinarySelect*.