Overcoming Black-box Attack Inefficiency with Hybrid and Dynamic Select Algorithms

Abhinay Shankar Belde, Rohit Ramkumar, Jonathan Rusert
Department of Computer Science
Purdue University, Fort Wayne
{belda01, ramkr01, jrusert}@pfw.edu

Abstract

Adversarial text attack research plays a crucial role in evaluating the robustness of NLP models. However, the increasing complexity of transformer-based architectures has dramatically raised the computational cost of attack testing, especially for researchers with limited resources (e.g., GPUs). Existing popular black-box attack methods often require a large number of queries, which can make them inefficient and impractical for researchers. To address these challenges, we propose two new attack selection strategies called Hybrid and Dynamic Select, which better combine the strengths of previous selection algorithms. Hybrid Select merges generalized BinarySelect techniques with GreedySelect by introducing a size threshold to decide which selection algorithm to use. Dynamic Select provides an alternative approach of combining the generalized Binary and GreedySelect by learning which lengths of texts each selection method should be applied to. This greatly reduces the number of queries needed while maintaining attack effectiveness (a limitation of BinarySelect). Across 4 datasets and 6 target models, our best method(sentence-level Hybrid Select) is able to reduce the number of required queries per attack up 25.82% on average against both encoder models and LLMs, without losing the effectiveness of the attack.

1 Introduction

Adversarial attacks have been useful for studying the robustness of NLP models. Black-box world-level attacks often follow a two-step approach, first selecting a word or token to modify and then modifying that word to cause the target NLP model to fail (with further constraints). Selection algorithms can play a pivotal role in the computational overhead of adversarial attacks. GreedySelect (Hsieh et al., 2019; Li et al., 2020b), a widely adopted method, iteratively evaluates each word in the text to determine its contribution to the classifier's decision. By removing/masking the word with the highest probability drop, GreedySelect achieves high precision. However, its word-by-word approach often results in high query counts, limiting its efficiency.

As NLP models continue to grow in size, high query counts can translate to attacks taking a large of amount of time per text. This inefficiency further affects researchers or users who do not have access to high-end GPU's to carry out attacks. Furthermore, an increased time in attack execution, reduces the ability for researchers to adequately test their models.

To address these challenges, BinarySelect (Ghosh and Rusert, 2025) adopted a divide-and-conquer strategy inspired by binary search. By partitioning the input into two segments and evaluating the probability change caused by excluding each segment, BinarySelect aimed to narrow down the search space when selecting an influential word. While BinarySelect helped reduced the number of queries compared to GreedySelect in some instances, it caused a drop in attack effectiveness.

Building on these foundations, this paper proposes hybrid approaches that integrate query-efficient methods like BinarySelect with GreedySelect's effectiveness. These strategies aim to balance attack efficacy and computational efficiency, addressing critical gaps identified in prior literature and offering a solution to improve adversarial attack performance.

Our research makes the following contributions:

1. **New Selection Algorithms (Hybrid and Dynamic Select).** We propose two new algorithms that extend BinarySelect and GreedySelect. *Hybrid Select* uses N-nary partitioning up to a threshold before switching to GreedySelect, while *Sentence-Level Hybrid* is a special case that partitions by sentence. *Dynamic Select* instead learns which N values are most effective for different text lengths, adapting

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 20565–20598
November 4-9, 2025

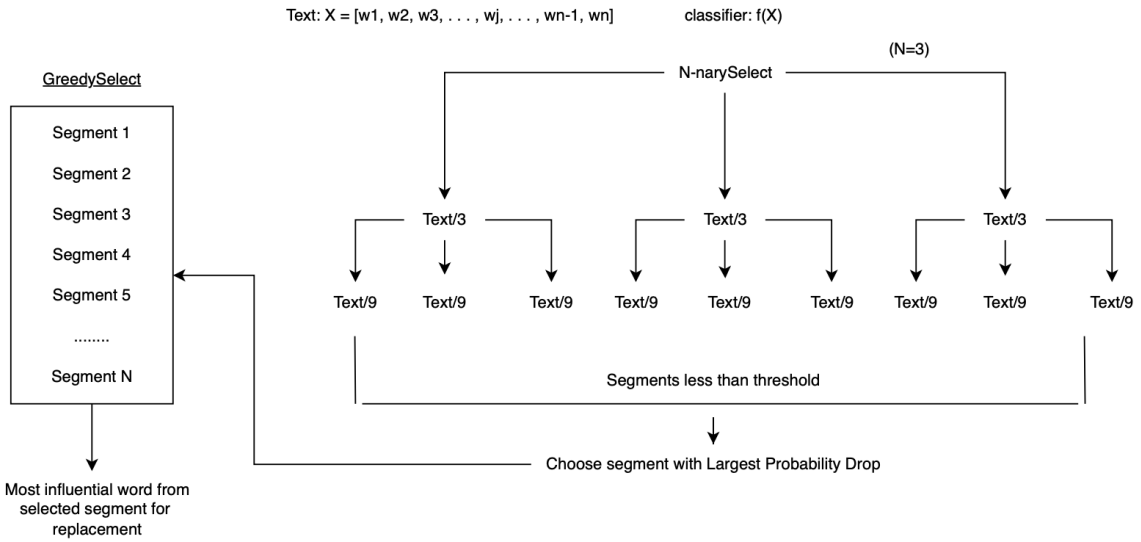

Figure 1: Visualization of HybridSelect. This figure illustrates the HybridSelect algorithm, which combines GreedySelect and N-narySelect techniques for efficient query optimization. The process begins with segmenting the input text X into smaller components. The GreedySelect method evaluates individual segments iteratively to determine their influence, identifying the most impactful word for replacement. Concurrently, N-narySelect partitions the text hierarchically, dividing it into n-segments (e.g., $N = 3$), and evaluates subqueries using a classifier $f(X)$. Probabilities are calculated, and the segment with the largest probability drop is selected. Segments below the threshold are finalized, ensuring an optimal balance between exploration and computational efficiency.

the strategy automatically. We also evaluate their combination (*Hybrid + Dynamic*), highlighting where it helps and where it fails. Across 4 datasets and 6 models, one variant of *Sentence-Level Hybrid* reduces query counts by up 25.82% (e.g., 484 → 362 queries (Table 1)) while preserving attack success. Unlike prior works that focus on improving the **replacement** stage, our contribution advances the **selection** step and generalizes to many greedy-based attacks (e.g., TextFooler, BERT-Attack, PWWS)[1].

2. **Evaluate Methods Across Multiple Datasets and Dimensions:** We evaluate our methods on adversarial attacks against both encoder models (DistilBert and DeBERTa) and LLMs (GPT2, GPTNeo, T5, and Llama3) across four standard adversarial attacks classification datasets of various sizes: IMDB, Yelp, AG News, and Rotten Tomatoes. Additionally, we conduct several ablation studies to find and demonstrate the strongest hyperparameters and confirm the effectiveness and efficiency of our methods.

Overall, our proposed methods improve efficiency over previous methods like GreedySelect while maintaining efficacy, which was not achieved by previous methods such as BinarySelect.

2 Proposed Approaches

Our proposed approaches focus on the domain of black-box attacks (sometimes noted as grey-box). That is, following previous research (Li et al., 2021a; Hsieh et al., 2019; Alzantot et al., 2018a; Gao et al., 2018), we assume information about the model, such as weights and architecture, are not known. The attack can pass a query to the model, which returns a prediction with a probability.

The goal of the attacker is to modify an input text, such that it causes a target model to fail, but retains its original meaning to humans (Li et al., 2020a; Garg and Ramakrishnan, 2020b). In this research, we focus on the domain of text classification. Often black-box word-level text attacks consist of 2 stages: selection and modification. In selection, the attacker finds the word or token that the classifier is relying on the most for its decision. In modification, the token found in the selection step is modified in such a way to keep the noted

[1]We share our code for reproducibility: https://github.com/08Abhinay/Hybrid-Dynamic-Select

goals. This can be character level modification or word level by replacing the token with less known synonyms. If the attack does not succeed after the modification, the selection method finds the next influential word and the process continues.

In this research, our objective is to improve the efficiency of attacks via the selection step by balancing the efficiency of BinarySelect and the attack effectiveness of GreedySelect. We first define GreedySelect and BinarySelect and then our own approaches.

2.1 Baseline Selection Algorithms

Baseline selection. *Greedy Select* (and its variations such as Importance Score) is one of the most widely used method for black-box text attacks (Formento et al., 2023; Jin et al., 2020; Li et al., 2020a; Garg and Ramakrishnan, 2020b; Ren et al., 2019; Hsieh et al., 2019; Li et al., 2019; Gao et al., 2018). Greedy Select removes one word at a time, queries the classifier, and keeps the word whose omission produces the largest drop in the target-class probability. Because it probes every token, it is highly effective but also query-intensive (one query per word just to find the first salient token). *Binary Select* (Ghosh and Rusert, 2025) eases this burden by applying a binary-search heuristic: the sentence is repeatedly split in half, and the half whose removal hurts the classifier most is subdivided until only a single word remains. This strategy cuts the query cost dramatically on long texts but—by discarding half of the context at each step—can overlook subtle but important words, leaving a small effectiveness gap relative to Greedy Select.

To bridge that gap we introduce *N-nary Select*, which generalises Binary Select from two to N segments. Starting with the full input X and its score $f(X)$, we partition X into N disjoint segments $\{X_1, \ldots, X_N\}$. For each segment we compute $\Delta_i = f(X) - f(X \setminus X_i)$ and recurse on the segment with the largest Δ_i. The recursion stops when a single-word segment is reached; that word is deemed most influential. By tuning N, the algorithm interpolates smoothly between the thoroughness of Greedy Select ($N = |X|$) and the speed of Binary Select ($N = 2$), allowing us to find a sweet-spot that is both query-efficient and consistently effective across text lengths and model architectures.

2.2 Proposed Selection Methods

Though *N-nary Select* is more flexible than BinarySelect, our experiments show, it still suffers from issues found in BinarySelect. Namely, on small texts, the model is better off leveraging GreedySelect as the division of texts can cause extra unnecessary queries. Thus, we propose *Hybrid Select* and *Dynamic Select* to address this.

2.2.1 *Hybrid Select*

Hybrid Select combines *N-nary Select* with GreedySelect by setting a length threshold t. The full algorithm can be found in Algorithms 2–9. Specifically, *Hybrid Select* first follows *N-nary Select* and continuously partitions the texts into N partitions. Once the length of the current partition falls below t, the algorithm transitions to word-level greedy exploration to pinpoint the most impactful words. The threshold t is set as a percentage of the original text length. For example, if the original text is 200 words in length and t is set to 10%, then once the partitioned texts contain 20 words or less, the algorithm switches to GreedySelect. This combination of N-nary segmentation and word-level precision balances computational efficiency with attack effectiveness. A visualization of this method can be found in Figure 1.

***Sentence Level Hybrid Select*:** Another issue with the *N-nary Select* algorithm is that splitting the text in N separate segments can cause sentences to split and meaning affected. Thus, we also propose a special case of the *Hybrid N-nary Select* methodology which splits the original text at the sentence level (Alg 7). This variation sets $N = S$, where S is the number of sentences in the original text. The method then follows the standard Hybrid procedure: if any sentence segment remains longer than the threshold t, it continues N-nary splitting; once the length drops below t, it switches to GreedySelect. In practice, this means the method first identifies the most influential sentence via *N-nary Select*, and then pinpoints the most influential words within that sentence via GreedySelect.

The hybrid nature of this approach enables it to leverage coarse-grained sentence segmentation for rapid narrowing, followed by fine-grained word exploration for precision. It is particularly effective in scenarios where sentence boundaries provide a natural structure to the input text.

2.2.2 *Dynamic Select*

We also propose an algorithm which leverages GreedySelect and *N-nary Select* on the texts they are the most effective on. That is, we find the optimal N value based on the length of the input text in words and then use this in the attack. As shown in Appendix D, these values are calculated from a separate validation set in the same dataset. Initially, the maximum input length in the validation set is identified. Based on this length, 5 length bins are created. For each length bins, the optimal N value is selected based on the minimum query count in the validation set. This mapping of length bins to the optimal N value is applied to the target data.

2.2.3 *Hybrid + Dynamic Select*

Finally, we aim to combine the strengths of both *Hybrid Select* and *Dynamic Select* into one methodology *Hybrid + Dynamic Select*. This method combines both algorithms by first using *Dynamic Select* to find the optimal N and then passing that N to the *Hybrid Select* methodology. Specifically, the algorithm will automatically choose the N value based on the binning rules shown in Algorithm 2.

2.3 Replacement Methodology

As our proposed improvements focus on the selection algorithm, we leverage a simple replacement algorithm leveraged by previous research (Ren et al., 2019). We leverage Wordnet (Miller, 1994) to obtain synonyms of selected words and try to replace the word and check the change in probability. We begin with the most similar synonym (as given by Wordnet) and continue checking every synonym until either the classifier fails or all have been checked. If the classifier did not fail, we choose the synonym that causes the highest drop in target label probability. We also perform experiments to see if utilizing a transformer model for masked language modeling replacement would improve the attack effectiveness or efficiency (similar to previous research (Li et al., 2020b; Garg and Ramakrishnan, 2020b)). These can be found in Section C.3.

3 Experimental Setup

To evaluate *Hybrid Select* and *Dynamic Select* in the adversarial attack setting, we test all noted selection methods as parts of attacks[2] against two encoder-based models Distilbert (Sanh et al., 2020) and DeBERTa (He et al., 2021) and 4 LLM models GPT2 (Radford et al., 2019), GPTNeo (Black et al., 2021), T5 (Raffel et al., 2020), and Llama3 (AI@Meta, 2024). These models are tested in various amounts across 4 standard attack datasets (Li et al., 2020b; Jin et al., 2020): three binary sentiment datasets (IMDB, Yelp Polarity, Rotten Tomatoes) and one multi-class news dataset AG News. IMDB and Yelp contain longer texts on average (215 and 157 words respectively) than AG News (43 words) and Rotten Tomatoes (20 words). We expect our methods to naturally perform better on longer texts due to the larger disadvantage GreedySelect has (it must perform $len(text)$ queries always at the beginning of an attack). Following previous attack research, we randomly sample 1000 examples from each dataset to attack and keep these 1000 consistent across combinations.

Metrics: We use the following metrics to evaluate our approaches in the attack:

1. Attack Success Rate (ASR) (Equation 1 - A standard metric in attack research. This helps measure the effectiveness of the attack.

$$\text{ASR} = \frac{\text{Original}_{\text{Acc.}} - \text{Attack}_{\text{Acc.}}}{\text{Original}_{\text{Acc.}}} \quad (1)$$

2. Average Queries - To measure increase in attack efficiency from our proposed methods, we measure the number of queries needed for an attack on average. These queries indicate how many calls to the target classifier are needed. We focus on only the successful attacks, following prior research (Liu et al., 2023).

3. Average Similarity. To ensure adversarial examples remain semantically close to the original text, we use cosine similarity between sentence embeddings from `all-mpnet-base-v2` (Reimers and Gurevych, 2020), a Sentence-BERT model fine-tuned on semantic textual similarity and NLI tasks. This model offers a strong balance of efficiency and accuracy, achieving high correlation with human judgments (Spearman $\rho \approx 0.865$) while being lightweight. Its robustness to surface-level edits and fast inference makes it ideal for measuring semantic preservation during attack-time.

4. Average Perturbation Amount - A similar motivation to similarity, the average perturbation amount measures how much of the original text was modified. As with the above, we calculate this for successful attacks.

[2]This research utilized some combination of Nvidia V100, A100, A10, and A30 GPUs. Each attack combination took roughly 30 minutes.

		DistilBERT (OA = 92)		**DeBERTa** (OA = 94)		**GPT-2** (OA = 94)		**GPT-Neo** (OA = 93)		**T5** (OA = 90)		**Llama3** (OA = 92)	
Selection Method	N	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**
Binary	-	98	484	71	372	90	652	98	583	98	480	98	547
Greedy	-	**99**	425	70	303	90	593	98	462	99	412	98	430
N-Nary	3	98	461	70	372	93	614	98	537	99	455	100	492
	6	98	569	71	447	92	831	99	643	99	496	98	581
Hybrid	2	98	427	70	333	99	507	98	485	99	391	98	458
	3	98	392	70	302	98	475	97	457	99	392	100	427
	6	98	398	70	360	99	533	99	555	99	422	98	504
Sent. Hybrid	2	98	368	71	303	98	430	97	456	98	384	97	459
	3	98	**362**	71	302	98	443	98	445	98	368	97	454
	6	98	373	**71**	**295**	98	**428**	97	**430**	98	**359**	97	436
Dynamic	-	98	456	70	384	98	552	98	553	99	429	98	497
Dynamic + Hybrid	-	**99**	395	71	312	69	463	73	467	99	376	98	**415**
Selection Method	N	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**
Binary	-	91	9	92	8	84	13	90	12	89	11	91	10
Greedy	-	91	7	92	7	85	12	91	8	90	8	92	7
N-Nary	3	91	8	91	8	84	14	90	11	89	11	91	10
	6	91	7	92	7	85	13	91	9	90	9	92	8
Hybrid	2	92	9	92	8	89	12	90	11	89	11	91	10
	3	92	8	91	8	89	11	90	11	89	11	91	10
	6	92	7	92	7	90	9	91	9	90	9	92	8
Sent. Hybrid	2	91	9	91	9	89	11	90	11	89	12	90	11
	3	91	9	90	8	89	11	90	11	89	11	90	10
	6	91	9	91	7	90	10	91	10	89	11	90	10
Dynamic	-	91	8	82	19	89	11	90	11	90	9	91	9
Dynamic + Hybrid	-	91	8	82	19	89	11	90	11	90	9	91	9

Table 1: Results for different methods on the IMDB dataset where ASR is attack success rate, AvgQ is the average queries by the successful attacks, Sim. is the similarity of text to the original, and Pert. is the percentage of text modified. A t of 10% was used for all Hybrid methods. **Bold** values indicate the best for that column, underline indicates second best.

Number of words to modify: BinarySelect introduced a parameter k which indicated how many words were allowed to be modified at most. In our experiments, we let k be equal to the number of words (or *ALL* in BinarySelect). We do perform more experiments on various restrictions of k in Section C.4.

4 Results and Discussions

Tables 1, 2, 3, and 4 present the evaluation of our attack strategies on IMDB, Yelp, AG News, and Rotten Tomatoes, respectively. For each dataset, we evaluate six target models—DistilBERT, DeBERTa, GPT-2, GPT-Neo, T5, and Llama3—and report four core metrics: attack success rate (ASR), average number of queries for successful attacks (AvgQ), semantic similarity to the original input (Sim.), and the percentage of tokens modified (Pert.). Alongside the baseline methods (Binary and Greedy), we evaluate fixed N-Nary splitting ($N \in \{2, 3, 6\}$) and our four novel algorithms: **Hybrid**, **Sentence-Level Hybrid**, **Dynamic-N**, and **Dynamic-N + Hybrid**. All Hybrid-based strategies apply a fixed pruning threshold of $t = 10\,\%$, which our preliminary experiments indicated provides a strong trade-off between efficiency and effectiveness. We explore these parameter choices more thoroughly in Sections C.1 and C.4. Our overarching objective remains to minimize queries and edits while reducing model confidence, and to demonstrate that our strategies scale reliably across model architectures, dataset lengths, and attack goals.

4.1 Attacks Versus Encoder Models

First we analyze the attacks against two encoder-style target models, **DistilBERT** and **DeBERTa**. On both models the Hybrid family consistently trims the number of queries without affecting the attack success. DistilBERT, shows

		DistilBert (OA = 97)		**DeBERTa** (OA = 99)		**GPT2** (OA = 96)		**GPTNeo** (OA = 97)		**T5** (OA = 97)	
Selection Method	N	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**
Binary	-	97	400	98	467	97	461	96	368	69	602
Greedy	-	**98**	329	95	462	96	**355**	97	323	74	**525**
N-Nary	3	97	374	95	469	96	426	96	342	71	605
	6	**98**	448	95	469	97	501	97	405	76	882
Hybrid	2	**98**	363	95	472	97	386	100	302	69	531
	3	**98**	346	**96**	497	96	367	96	289	69	543
	6	**98**	331	**96**	570	97	433	97	342	72	795
Sent. Hybrid	2	97	327	95	**403**	95	380	95	312	68	549
	3	97	**319**	**96**	511	95	382	95	307	69	559
	6	97	325	**96**	639	95	383	95	307	68	560
Dynamic	-	97	378	**96**	525	96	428	96	337	68	581
Dynamic+ Hybrid	-	97	**319**	**96**	462	96	370	96	**282**	70	543
Selection Method	N	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**
Binary	-	86	13	81	18	84	15	87	13	76	22
Greedy	-	86	11	82	15	86	12	87	11	76	20
N-Nary	3	86	12	81	18	85	14	87	12	76	22
	6	87	10	83	15	86	12	88	11	76	20
Hybrid	2	87	12	81	18	84	15	87	13	76	22
	3	87	12	81	17	85	14	87	12	76	21
	6	88	10	83	15	86	12	88	10	77	19
Sent. Hybrid	2	87	12	81	18	84	14	86	13	74	23
	3	87	12	81	18	84	15	86	12	74	23
	6	87	12	81	17	85	14	87	11	75	22
Dynamic	-	86	12	80	18	85	14	87	12	76	21
Dynamic + Hybrid	-	86	12	80	18	85	13	87	12	76	21

Table 2: Results for different methods on the Yelp dataset where ASR is attack success rate, AvgQ is the average queries by the successful attacks, Sim. is the similarity of text to the original, and Pert. is the percentage of text modified. A t of 10% was used for all Hybrid methods. **Bold** values indicate the best for that column, underline indicates second best.

more vulnerability to attacks and the largest query reductions.For IMDB, the sentence-level Hybrid with $N = 3$ drops the average query budget from 425 to 362 while holding the success rate at 98%. Similarly, for Yelp, Hybrid $N = 3$ (or Dynamic + Hybrid) reduces the queries to 319 with a 97% success rate.

We find efficiency improvements against DeBERTa as well, despite its higher original accuracy. For example, a sentence-level Hybrid with $N = 2$ reduces queries by about 13 % on Yelp, and on AG News the word-level Hybrid with $N = 3$ reduces the average down from 154 to 138—again with little change in ASR. Even on Rotten Tomatoes, where baseline queries are already low, Sentence-level Hybrid requires only 49–54 requests versus the Greedy baseline's 60, confirming the strength of the method across across text lengths and model sizes.

Examining the similarity and perturbation scores, we find that the efficiency gains come at little cost to text quality. On IMDB, Greedy achieves the highest similarity score for DistilBERT (91 %) with only 8 % of tokens edited; Hybrid $N = 3$ follows within a single point—90 % similarity—while touching 10 % of the text. DeBERTa mirrors that story: Greedy peaks at 92 % similarity and an 8 % perturbation rate, whereas Hybrid $N = 3$ stays level on similarity and rises only to 10 % edited

		DistilBert (OA = 95)		**DeBERTa** (OA = 96)		**GPT2** (OA = 95)		**GPTNeo** (OA = 93)		**T5** (OA = 95)		**Llama3** (OA = 96)	
Selection Method	N	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**
Binary	-	73	153	77	171	71	180	71	167	67	191	66	184
Greedy	-	73	153	78	154	72	156	71	153	67	171	67	155
N-Nary	3	72	161	77	163	71	171	70	157	65	182	66	176
	6	**76**	253	**79**	241	75	264	75	243	72	297	69	259
Hybrid	2	72	140	77	142	71	150	70	139	67	162	66	153
	3	72	**138**	77	**138**	71	**146**	70	**134**	66	159	66	152
	6	**76**	220	78	209	75	229	75	215	71	262	69	228
Sent. Hybrid	2	72	160	77	166	71	171	70	158	73	179	66	182
	3	73	164	77	162	71	168	71	162	66	180	66	175
	6	73	172	76	169	72	181	71	169	67	197	66	185
Dynamic	-	72	169	77	167	71	171	70	159	66	182	66	176
Dynamic + Hybrid	-	72	141	77	140	71	**146**	70	135	67	**157**	66	**151**
Selection Method	N	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**
Binary	-	74	27	73	29	75	30	77	28	73	31	72	31
Greedy	-	74	26	74	28	75	27	77	26	74	29	72	28
N-Nary	3	74	27	74	29	75	29	77	27	73	30	72	30
	6	74	26	74	27	76	28	77	25	75	29	72	28
Hybrid	2	74	27	72	29	75	30	77	27	73	30	72	31
	3	74	28	73	30	75	29	77	27	74	30	72	31
	6	74	27	73	27	76	8	78	25	75	28	72	29
Sent. Hybrid	2	74	27	72	29	75	29	77	27	74	29	72	31
	3	74	28	73	28	75	29	78	27	73	31	72	31
	6	75	27	73	28	75	28	78	26	75	29	72	29
Dynamic	-	73	32	73	22	75	29	77	27	74	29	72	30
Dynamic + Hybrid	-	73	32	74	33	75	29	77	27	73	30	72	30

Table 3: Results for different methods on the AGNews datasetwhere ASR is attack success rate, AvgQ is the average queries by the successful attacks, Sim. is the similarity of text to the original, and Pert. is the percentage of text modified. A t of 10% was used for all Hybrid methods. **Bold** values indicate the best for that column, underline indicates second best.

tokens. In short, the encoders let us trade a modest 2 % increase in edits for a double-digit drop in queries.

4.2 Attacks Versus LLMs

We find similar efficiency improvements against the LLMs—**GPT-2**, **GPT-Neo**, **T5**, and **Llama3**. However, the gains depend on both model robustness and dataset length. On the sprawling IMDB corpus, the Hybrid agenda is outperforms the rest of the models: for GPT-2, word-level Hybrid with $N = 3$ reduces queries by 20 % (593→475) and lifts the success rate from 90 % to 98 %. Furthermore, it boosts similarity from 83 % to roughly 88–89 % while reducing perturbations from 15 % to 13 %. GPT-Neo and T5 show milder query reductions, yet the sentence-level Hybrid with $N = 6$ still shaves about fifty requests apiece and leaves similarity unchanged in the high-80s. Llama3 is particularly revealing: its baseline already sits at 98 % success and 90 % similarity, yet Hybrid $N = 3$ nudges success to a perfect 100 % while keeping similarity above 89 % and slightly lowering the number of edits.

We find success for Yelp as well depending on the model examined. Greedy remains a strong baseline for GPT-2, but on GPT-Neo the Hybrid $N = 3$ variant reduces queries by 11 % (323→289) with little to no loss of ASR and with similarity still around 86 %. Dynamic + Hybrid sees greater reductions, maintaining the same similarity and a modest 14 % perturbation rate. T5 is the single hold-out: its success plateau of ∼74 % appears insensitive to selection strategy; however, Hybrid does not worsen quality, holding similarity near 75 % and perturbation near 23 %.

We find that the length plays a role in efficiency gain. On short news snippets (AG News)

		DistilBert (OA = 83)		**DeBERTa** (OA = 91)		**GPT2** (OA = 85)		**GPTNeo** (OA = 84)		**T5** (OA = 89)	
Selection Method	N	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**	**ASR**	**AvgQ**
Binary	-	78	54	90	64	95	53	95	51	89	63
Greedy	-	78	50	91	60	95	49	95	48	90	58
N-Nary	3	78	52	90	61	95	50	95	48	90	61
	6	78	58	92	68	94	55	95	54	92	69
Hybrid	2	78	38	89	**49**	95	39	95	38	89	**48**
	3	77	38	89	**49**	95	**38**	95	**36**	89	**48**
	6	78	42	**93**	54	94	40	95	39	91	56
Sent. Hybrid	2	78	38	90	65	95	55	95	53	90	65
	3	77	38	90	65	95	53	95	51	90	64
	6	78	42	92	71	94	58	95	57	91	72
Dynamic	-	78	50	91	63	95	51	95	48	90	61
Dynamic + Hybrid	-	**78**	**37**	91	63	95	38	95	**36**	90	**48**
Selection Method	N	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**	**Sim.**	**Pert.**
Binary	-	80	16	75	19	79	17	80	16	77	19
Greedy	-	79	17	76	19	80	16	80	15	77	18
N-Nary	3	81	16	76	19	79	17	80	16	77	19
	6	81	15	77	18	80	14	81	14	78	18
Hybrid	2	80	15	75	19	78	17	80	16	77	19
	3	80	15	74	20	79	17	79	15	77	19
	6	80	14	76	18	80	14	81	14	77	18
Sent. Hybrid	2	79	16	75	19	78	17	80	16	77	19
	3	80	15	76	19	79	16	80	16	77	19
	6	80	15	76	18	80	15	81	14	78	18
Dynamic	-	82	14	76	20	79	16	80	15	78	18
Dynamic + Hybrid	-	82	14	76	20	79	16	80	15	78	18

Table 4: Results for different methods on the RottenTomatoes dataset where ASR is attack success rate, AvgQ is the average queries by the successful attacks, Sim. is the similarity of text to the original, and Pert. is the percentage of text modified. A t of 10% was used for all Hybrid methods. **Bold** values indicate the best for that column, underline indicates second best.

and single-sentence reviews (Rotten Tomatoes) every method achieves success in under 200 and 60 queries respectively; here the Hybrid advantage is measured in tens rather than hundreds of queries. The quality metrics are also closer. AG News similarity is in the low-70s for all models, with perturbations around 25–30 %. Hybrid $N = 3$ matches Greedy's similarity while removing one or two points off the perturbation rate on several models, suggesting its pruning step discards non-essential words. Rotten Tomatoes sees stronger results: similarities range 72–79 % and perturbations sit between 16 % and 22 %. Dynamic + Hybrid tops DistilBERT at 79 % similarity while tying Greedy for the lowest perturbation rate (16 %).

4.3 Overall Obversations

Across model families the pattern is consistent: the encoders are easiest to attack and preserve the quality of the text (high-80s to low-90s similarity with $\leq$10 % edits), GPT-style decoders trail by a few points but improve once Hybrid pruning is enabled, and T5 lags modestly yet never degrades further with the new strategies. Importantly, no Hybrid variant inflates perturbation beyond the baseline range; where similarity dips—such as Dynamic on DeBERTa + IMDB—it does so in tandem with a visibly higher 21 % edit rate, flagging an edge case rather than a systemic flaw.

Overall we find that *Hybrid Select*, with a simple

Method	N	Original Accuracy	Attack Accuracy	ASR	Avg Query	Avg Query (For Atk Success)
N-Nary Select	3	92	1.5	98	**475**	**461**
	6		1.3	98	585	569
	12		1.3	98	706	691
	24		**0.4**	**99**	747	732

Table 5: N-Nary results for larger N on the IMDB Dataset. **Bold** indicates the best value for each column, underline indicates second best.

10 % pruning heuristic, applied either at the word level with $N = 3$ or at the sentence level for very long documents, consistently lowers the query cost of gradient-free attacks without sacrificing meaning. For DistilBERT and DeBERTa the reduction is typically 10–15 %, while for the larger LLMs it can exceed 20 % on the largest corpora and still deliver equal or higher success rates. At the same time the semantic-similarity scores stay within one point of the best baselines, and perturbation rates remain flat—confirming that the extra efficiency comes from smarter search, not heavier rewrites.

4.4 Additional Result Comparisons to previous Greedy Based Attacks

We further compare our proposed methods to 2 greedy based attacks PWWS (Ren et al., 2019) and CLARE (Li et al., 2021b). We find our selection methods to be far more efficient, requiring 1/4 or less amount of queries but less subtle than these previous attacks, using 2 to 4 times amount of perturbation rates. The full table of results and further discussion can be found in Appendix B.

5 Ablation Studies

We perform further experiments to better understand our proposed methods. Each experiment expands on specific hyper-parameters or choices in the overall algorithm. Due to space, the full results of the studies can be found in Appendix C. We provide a short summary of the findings.

Effect of N in N-Nary Algorithm: We explore multiple increasing N values of 12 and 24 on the IMDB dataset against DistilBert (seen in Table 5). Keeping in mind N, we see a slight increase in overall ASR (from 98 to 99) with N = 24. We find for N = 3, the lowest Avg Query count of 475 however, followed by 6 (585), 12 (706), and 24 (747).

Impact of Split Threshold t for Hybrid N-nary: We analyze various split thresholds for the Hybrid algorithm beyond 10%, including 5%, 20%, 30%, and 40% on the IMDB dataset against DistilBert. We find that 5% and 10% provide the best query counts across various N choices, with an increase in query count as the threshold rises.

Impact of Word Replacement Strategies: We test our algorithm with a BERT mask and replace method used by other attacks, but found no significant improvement in the attack effectiveness or efficiency.

Impact of k: We also explore how various the hyper-parameter k (introduce by BinarySelect) affects the overall results. We find consistent observations with previous research. As k increases, average query count increases as does ASR.

6 Conclusion

Our study introduces a family of selection strategies—Hybrid, Sentence-Level Hybrid, Dynamic-N, and Dynamic-N + Hybrid—that strike an effective balance between efficiency and semantic fidelity in black-box text attacks. Across all four datasets and six model types, our methods consistently reduce query counts, sometimes by more than 20%, without compromising attack success or text quality.

In particular, we find that word-level Hybrid with N=3 offers the best overall trade-off, especially for long-form datasets like IMDB, while sentence-level variants perform better on lengthy inputs such as Yelp. Even on compact datasets like AG News and Rotten Tomatoes, Hybrid strategies retain their advantage, trimming redundant edits while preserving meaning.

Overall, these results validate that a simple, fixed pruning heuristic—anchored in flexible, recursive search—can deliver scalable and transferable improvements across diverse models and datasets. Our methods offer a principled yet practical approach for adversarial attacks where query efficiency and semantic clarity are both essential.

7 Limitations

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

1. **Lack of Human Validation in Replacement Evaluation:** While our results demonstrate the effectiveness of N-Nary Select, we did not incorporate human validation to assess the relevance of selected words or phrases. This limitation stems from the focus on classifier feedback rather than human interpretability. Classifiers often prioritize features that differ from human-perceived importance, making it challenging to evaluate the semantic quality of replacements directly. Future work could explore incorporating human evaluations or applying these methods to tasks where human feedback is readily available, providing additional insights into the interpretability of selected replacements.

2. **Focus on Selection Efficiency Over Replacement Diversity:** Our algorithms prioritize selection efficiency by minimizing query counts and achieving higher Attack Success Rates (ASR). However, this focus may come at the expense of exploring diverse replacement options. Certain adversarial attacks might benefit from introducing more diverse perturbations to maximize model degradation. Future researchers could expand on our work by balancing selection efficiency with replacement diversity to explore its impact on attack robustness.

8 Ethical Considerations

Ethical considerations are critical in adversarial research due to potential misuse. While our methods could be exploited for harmful purposes, such as bypassing security systems or spreading misinformation, they also offer significant benefits. Our work provides valuable insights into attack strategies that can guide the development of stronger defenses.

By making our code publicly available, we encourage transparency and collaboration in the research community. This enables researchers with limited resources to contribute to advancements in both attack and defense strategies.

Although focused on adversarial attacks, our findings support defense-oriented research by helping design models resilient to such attacks. We advocate for the responsible use of our work to enhance model robustness and security.

References

AI@Meta. 2024. The llama 3 herd of models.

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

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018b. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

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.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Diyi Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 31–36.

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.

Shankar Garg and Ganesh Ramakrishnan. 2020a. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6174–6181.

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

Shatarupa Ghosh and Jonathan Rusert. 2025. BinarySelect to improve accessibility of black-box attack research. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10960–10976, Abu Dhabi, UAE. Association for Computational Linguistics.

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*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

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.

Robin Jia, Aditi Raghunathan, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 4145–4155.

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.

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. 2021a. 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.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021b. 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. 2020a. 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.

Siyu Li, Yuan Zhang, Lei Hou, Juanzi Liu, and Zhiyuan Li. 2020b. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6193–6202.

Han Liu, Zhi Xu, Xiaotong Zhang, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. 2023. Hqa-attack: toward high quality black-box hard-label adversarial attack on text. *Advances in Neural Information Processing Systems*, 36:51347–51358.

George A. Miller. 1994. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525.

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.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912.

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.

Naomi Saphra and Adam Lopez. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756.

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.

Haotian Wang, Yuxuan Wang, and Rui Ren. 2020. Natural language adversarial defense through synonym encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6189–6198.

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

Qian Zhu, Feng Shi, Junbo Zhao, and Guoqing Zhang. 2021. Towards improving adversarial training of nlp models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 690–701.

Algorithm 1: Notation

Classifier: $f(\cdot)$ returns per-class probabilities.
Inputs: text x; target/attack class index y; dataset ID ds.
Query/score: $f(x)[y]$ denotes the probability of class y for x.
Branching: N (N-ary split factor); $\mathcal{T}$ (search tree over spans).
Budget: k (max replacements; -1 means unlimited).
Replace source: `replace` $\in \{\text{wordnet}, \text{bert}\}$.
Counters: q (total queries), Δq (queries this step).
Misc: p_0 (initial $f(x)[y]$); x^{cur} (current text).

Algorithm 2: DynN (Automatic N Selection)

Input: length $L = |x|$, dataset id ds
Output: split factor N
Load dataset-specific length bins $\{([\ell_i, u_i], N_i)\}_i$ for ds
for each bin $([\ell, u], N)$ **do**
  if $\ell < L \leq u$ **then**
    return N
  end if
end for
return 2 {fallback}

Algorithm 3: WordNetReplace (single-position synonym trial)

Input: f, x, y, p_0, i {index i to try replacing}
Output: success flag s, updated text x^{cur}, prob p, queries Δq
$w \leftarrow$ token at position i in x; $S \leftarrow$ WordNet synonyms of w
if $S = \emptyset$ **then**
  return (**false**, x, p_0, 0)
end if
best_prob $\leftarrow p_0$; best_x $\leftarrow x$; $\Delta q \leftarrow 0$
for each $s \in S$ **do**
  $x' \leftarrow x$ with token i replaced by s
  $p \leftarrow f(x')[y]$; $\Delta q \leftarrow \Delta q + 1$
  if $\arg\max f(x') \neq y$ **then**
    return (**true**, x', p, Δq)
  end if
  if $p <$ best_prob **then**
    best_prob $\leftarrow p$; best_x $\leftarrow x'$
  end if
end for
return (**false**, best_x, best_prob, Δq)

Algorithm 4: BERTReplace (single-position masked-LM trial)

Input: f, x, y, p_0, i; masked LM MLM; top-M candidates
Output: success flag s, updated text x^{cur}, prob p, queries Δq
Get top-M candidate tokens $S \leftarrow \mathrm{MLM}(x, i)$ {mask token i}
if $S = \emptyset$ **then**
    return (**false**, x, p_0, 0)
end if
best_prob $\leftarrow p_0$; best_x $\leftarrow x$; $\Delta q \leftarrow 0$
for each $s \in S$ **do**
    $x' \leftarrow x$ with token i replaced by s
    $p \leftarrow f(x')[y]$; $\Delta q \leftarrow \Delta q + 1$
    if $\arg\max f(x') \neq y$ **then**
        return (**true**, x', p, Δq)
    end if
    if $p <$ best_prob **then**
        best_prob $\leftarrow p$; best_x $\leftarrow x'$
    end if
end for
return (**false**, best_x, best_prob, Δq)

Algorithm 5: N_Nary_Select_Iter (segment scoring)

Input: $f, x, y, N, \mathcal{T}$
Output: token index i (if leaf), queries Δq, updated tree $\mathcal{T}$
$\Delta q \leftarrow 0$
if root of $\mathcal{T}$ not set **then**
    create root with span covering all of x and prob $f(x)[y]$; $\Delta q \leftarrow \Delta q + 1$
end if
$u \leftarrow$ lowest-probability non-explored node in $\mathcal{T}$
Split u's span into N equal subspans $\{s_1, \ldots, s_N\}$
for each subspan s_j **do**
    form $x^{(-s_j)}$ by dropping s_j from x; compute $p_j \leftarrow f(x^{(-s_j)})[y]$
    $\Delta q \leftarrow \Delta q + 1$; attach/refresh child node for s_j with prob p_j
end for
Choose child c with the largest drop $f(x)[y] - p_j$
if c is a single-token span **then**
    mark c explored; **return** (its token index, $\Delta q, \mathcal{T}$)
else
    set $u \leftarrow c$ and **repeat** the split/score steps until a single-token child is chosen
    return (token index of final child, $\Delta q, \mathcal{T}$)
end if

Algorithm 6: N_Nary_Select_Segment (descend until size threshold)

Input: $f, x, y, N, \mathcal{T}, \tau$ {τ = split_threshold_percentage of $|x|$}
Output: a promising segment $[a, b]$, queries Δq, updated $\mathcal{T}$
$\Delta q \leftarrow 0$
if root of $\mathcal{T}$ not set **then**
  create root with span covering x and prob $f(x)[y]$; $\Delta q \leftarrow \Delta q + 1$
end if
$u \leftarrow$ lowest-probability non-explored node in $\mathcal{T}$
while span_length(u) $> \tau \cdot |x|$ **do**
  Split u into N equal subspans $\{s_1, \ldots, s_N\}$
  for each subspan s_j **do**
    $p_j \leftarrow f(x^{(-s_j)})[y]$; $\Delta q \leftarrow \Delta q + 1$; update child node prob
  end for
  $u \leftarrow \arg\min_{s_j} p_j$ {child yielding largest drop}
end while
return (segment of u, $\Delta q, \mathcal{T}$)

Algorithm 7: HybridSentence_WNR (Sentence-level → Token-level)

Input: $f, x, y, ds, m, N_{man}, k,$ `replace`
Output: success flag, x^{cur}, q, final prob p
$p_0 \leftarrow f(x)[y]$; $q \leftarrow 1$; $x^{cur} \leftarrow x$
Tokenize x into sentences $\{s_j\}$
In batch, for each j: compute $p_j \leftarrow f(x$ without $s_j)[y]$; $q \leftarrow q + |\{s_j\}|$
Pick $s^{\leftarrow \arg\min_j p_j}$ {largest drop}
$N \leftarrow$ **if** $m =$ auto **then** $\mathrm{DynN}(|s^{|,ds)}$ **else** N_{man}
Initialize $\mathcal{T}$ rooted at span covering s
while true **do**
  $(i, \Delta q, \mathcal{T}) \leftarrow \mathrm{N\_Nary\_Select\_Iter}(f, x^{cur}, y, N, \mathcal{T})$; $q \leftarrow q + \Delta q$
  $(s, x^{cur}, p, \Delta q) \leftarrow \mathrm{WordNetReplace}(f, x^{cur}, y, p_0, i)$; $q \leftarrow q + \Delta q$; $k \leftarrow k - 1$
  if $s =$ **true then**
    return (**true**, x^{cur}, q, p)
  end if
  if $k = 0$ **or** no unexplored node in $\mathcal{T}$ **then**
    return (**false**, x^{cur}, q, p_0)
  end if
end while

Algorithm 8: HybridDynN_WNR (Token-level Hybrid)

Input: $f, x, y, ds, m, N_{man}, k,$ `replace`
Output: success flag, x^{cur}, q, final prob p
$N \leftarrow$ **if** $m =$ auto **then** $\mathrm{DynN}(|x|, ds)$ **else** N_{man}
$p_0 \leftarrow f(x)[y]$; $q \leftarrow 1$; $x^{cur} \leftarrow x$; $\mathcal{T} \leftarrow \emptyset$
while true **do**
  $(i, \Delta q, \mathcal{T}) \leftarrow \mathrm{N\_Nary\_Select\_Iter}(f, x^{cur}, y, N, \mathcal{T})$; $q \leftarrow q + \Delta q$
  {(Optional) extra greedy scoring around i using batched token removals to refine choice}
  $(s, x^{cur}, p, \Delta q) \leftarrow \mathrm{WordNetReplace}(f, x^{cur}, y, p_0, i)$; $q \leftarrow q + \Delta q$; $k \leftarrow k - 1$
  if $s =$ **true then**
    return (**true**, x^{cur}, q, p)
  end if
  if $k = 0$ **or** no unexplored node in $\mathcal{T}$ **then**
    return (**false**, x^{cur}, q, p_0)
  end if
end while

Algorithm 9: HybridPure_WNR (No Sentence Stage / Pure Hybrid)

Input: $f, x, y, ds, m, N_{man}, k,$ `replace`, τ {τ=threshold fraction}
Output: success flag, x^{cur}, q, final prob p
$N \leftarrow$ **if** $m =$ auto **then** $\mathrm{DynN}(|x|, ds)$ **else** N_{man}
$p_0 \leftarrow f(x)[y]$; $q \leftarrow 1$; $x^{cur} \leftarrow x$; $\mathcal{T} \leftarrow \emptyset$
while true **do**
  $([a, b], \Delta q, \mathcal{T}) \leftarrow \mathrm{N\_Nary\_Select\_Segment}(f, x^{cur}, y, N, \mathcal{T}, \tau)$; $q \leftarrow q + \Delta q$
  if $[a, b] = \emptyset$ **then**
    return (**false**, x^{cur}, q, p_0)
  end if
  {Greedy per-token scoring within chosen segment (batched removal)}
  For each $t \in [a, b]$: form $x^{(-t)}$ by dropping token t; compute $p_t \leftarrow f(x^{(-t)})[y]$
  $q \leftarrow q + (b - a + 1)$; $i \leftarrow \arg\min_{t \in [a,b]} p_t$
  if `replace` = wordnet **then**
    $(s, x^{cur}, p, \Delta q) \leftarrow \mathrm{WordNetReplace}(f, x^{cur}, y, p_0, i)$
  else
    $(s, x^{cur}, p, \Delta q) \leftarrow \mathrm{BERTReplace}(f, x^{cur}, y, p_0, i)$
  end if
  $q \leftarrow q + \Delta q$; $k \leftarrow k - 1$
  if $s =$ **true then**
    return (**true**, x^{cur}, q, p)
  end if
  if $k = 0$ **or** no unexplored node in $\mathcal{T}$ **then**
    return (**false**, x^{cur}, q, p_0)
  end if
end while

A Related Work

Adversarial text attacks have emerged as a critical area of research for testing the robustness of Natural Language Processing (NLP) models against manipulative inputs. These attacks, spanning various levels such as character, word, phrase, sentence, and multi-level manipulations, aim to exploit vulnerabilities in NLP systems. Early work, such as HotFlip (Ebrahimi et al., 2018), introduced white-box attacks by leveraging gradient-based strategies to replace words and deceive classifiers. White-box attacks, which have complete access to model information, including weights and architecture (Sadrizadeh et al., 2022; Wang et al., 2022), are efficient in identifying impactful perturbations due to their comprehensive model knowledge. However, black-box attacks have gained prominence for their practicality in real-world scenarios where only confidence levels or predicted labels are accessible (Le et al., 2022; Jin et al., 2020). Foundational work, such as *Generating Natural Language Adversarial Examples* (Alzantot et al., 2018b), pioneered black-box techniques by utilizing word saliency scores to craft adversarial examples, demonstrating the feasibility of attacking models without direct access to their internals.

At the character level, adversarial methods manipulate individual characters to disrupt tokenization processes. Techniques include adding or removing whitespace (Gröndahl et al., 2018), replacing characters with visually similar alternatives (Eger et al., 2019), or shuffling characters (Li et al., 2019). Word-level attacks focus on synonym replacements, leveraging resources like Word Embeddings (Hsieh et al., 2019), WordNet (Ren et al., 2019), and Mask Language Models (Li et al., 2020a) to identify substitutes that maintain semantic coherence while deceiving classifiers. Phrase-level approaches replace multiple consecutive words (Deng et al., 2022; Lei et al., 2022), while sentence-level techniques generate adversarial text to exploit model weaknesses (Ribeiro et al., 2018; Zhao et al., 2018). Multi-level attacks combine these methods for more comprehensive perturbations (Formento et al., 2023). Though our research focuses on word-level attacks, the methodology can extend to character or phrase-level attacks seamlessly.

Transformer-based models like BERT have revealed new vulnerabilities in NLP systems. For example, *BAE: BERT-Based Adversarial Examples for Text Classification* (Garg and Ramakrishnan, 2020a) and *BERT-ATTACK* (Li et al., 2020b) highlight how contextual embeddings can be exploited. These attacks underscore the need for enhanced defenses, particularly as adversarial attacks evolve. Defensive strategies like adversarial training have emerged as a popular countermeasure, incorporating adversarial examples during training to bolster robustness. For instance, *Natural Language Adversarial Defense through Synonym Encoding* (Wang et al., 2020) and *Towards Improving Adversarial Training of NLP Models* (Zhu et al., 2021) demonstrate promising advancements. However, challenges persist in achieving generalizable defenses that can withstand novel attacks.

To further address robustness, behavioral testing frameworks such as *Beyond Accuracy: Behavioral Testing of NLP Models with CheckList* (Ribeiro et al., 2020) have been introduced. These methodologies systematically evaluate model performance under diverse adversarial scenarios, emphasizing the importance of comprehensive testing to uncover potential weaknesses. Nevertheless, achieving a balance between attack success rates, query efficiency, and interpretability remains a significant challenge. Research like *Pathologies of Neural Models Make Interpretations Difficult* (Saphra and Lopez, 2018) highlights the difficulty of interpreting adversarial perturbations, while *Certified Robustness to Adversarial Word Substitutions* (Jia et al., 2019) explores trade-offs between computational efficiency and robustness guarantees.

B Comparison with Previous Greedy Attacks

We additionally compare our proposed selection methods with two other attacks which leverage a variation of GreedySelect: PWWS (Ren et al., 2019) and CLARE (Li et al., 2021a). The results can be found in Table 6. As can be observed, both PWWS and CLARE use a significantly higher amount of queries. For example on IMDB, PWWS uses over 4 times as many queries as our most efficient method (Sentence Level Hybrid) and CLARE uses 81 times as many. CLAREs higher use of queries allow for a higher ASR and Similarity and PWWS achieves higher similarity and lower perturbation rate on average. This highlights another tradeoff between efficiency and subtlety of attack, as our methods require 2 to 4 times more perturbation rates.

			DistilBert				DeBERTa			
	Method	N	**ASR**	**AvgQ**	**Sim.**	**Pert.**	**ASR**	**AvgQ**	**Sim.**	**Pert.**
IMDB	PWWS	-	100	1498	96	5	56	1391	96	4
	CLARE	-	100	29460	98	1	51*	22637	98	1
	Hybrid	3	98	392	91	8	70	302	91	8
	Sent. Hybrid	3	98	362	90	9	71	302	90	9
	Dynamic	-	98	456	90	8	70	384	82	19
	Dynamic + Hybrid	-	99	395	90	8	71	312	82	19
Yelp	PWWS	-	99	934	94	6	97	985	92	8
	CLARE	-	100	20194	97	3	99	22107	96	3
	Hybrid	2	98	363	87	12	95	472	81	18
	Sent. Hybrid	2	97	327	87	12	95	403	81	18
	Dynamic	-	97	378	86	12	96	525	80	18
	Dynamic + Hybrid	-	97	319	86	12	96	462	80	18
AG News	PWWS	-	62	338	88	20	63	333	89	22
	CLARE	-	97	8113	91	8	92	8541	91	8
	Hybrid	3	72	138	74	28	77	138	73	30
	Sent. Hybrid	3	73	164	74	28	77	162	73	28
	Dynamic	-	72	169	73	32	77	167	73	22
	Dynamic + Hybrid	-	72	141	73	32	77	140	74	33
Rotten Tomatoes	PWWS	-	86	139	86	12	81	147	84	13
	CLARE	-	100	1605	88	7	100	1668	87	7
	Hybrid	2	78	38	80	15	89	49	75	19
	Sent. Hybrid	2	78	38	79	16	90	65	75	19
	Dynamic	-	78	50	82	14	91	63	76	20
	Dynamic + Hybrid	-	78	37	82	14	91	63	76	20

Table 6: Results for different methods on the noted datasets where ASR is attack success rate, AvgQ is the average queries by the successful attacks, Sim. is the similarity of text to the original, and Pert. is the percentage of text modified. A t of 10% was used for all Hybrid methods. **Bold** values indicate the best for that column, underline indicates second best. * - Clare - IMDB was limited to 25000 query budget due to hardware constraints.

C Ablation Studies

To better understand the individual contributions of each component in our proposed attack strategies, we performed a series of ablation experiments. These experiments isolate the effects of selection algorithms. In the following, we provide a detailed analysis of the results.

C.1 Effect of N in N-Nary Algorithm

The N-Nary Select approach introduces hierarchical splitting, and we assess the impact of varying n (e.g., 3-Nary, 6-Nary, 12-Nary, 24-Nary). Lower values of n balance efficiency and effectiveness, achieving high attack success rates (ASR) with moderate query counts. For instance, on the IMDB dataset, 3-Nary Select achieves an ASR of 98% with an average query count of 475, while 24-Nary achieves a similar ASR with 747 queries. This

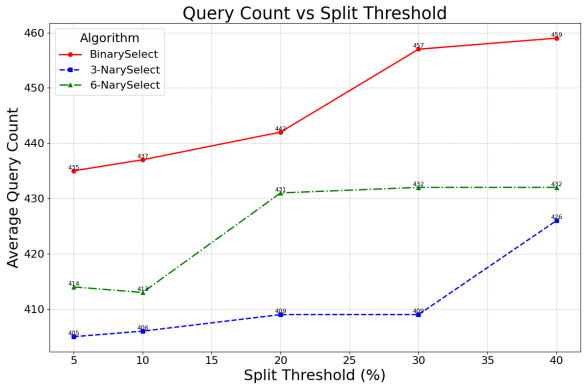

Figure 2: Query count versus Split Threshold t for Hybrid N-nary.

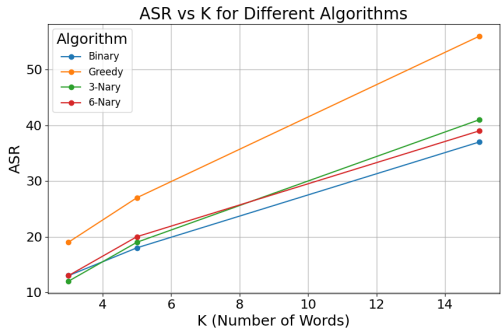

Figure 3: k (number of words allowed to be modified) versus ASR for base N-nary Select Algorithms. Results are consistent with previous BinarySelect work.

highlights a trade-off between finer granularity and computational overhead.

On the AGNews dataset, where text length is shorter, higher n values show diminishing returns. For example, 6-Nary achieves no significant improvement in ASR over 3-Nary but increases the average query count by over 17%. These results underscore the importance of selecting an appropriate n based on dataset characteristics.

Among the algorithms evaluated, the 3-nary Select consistently achieves the best balance between query efficiency and attack accuracy. It performs well at lower split thresholds, with fewer queries required to achieve successful attacks compared to Binary and 6-nary searches.

C.2 Impact of Split Threshold t for Hybrid N-nary

We analyze the impact of varying split thresholds on the performance of the hybrid Select algorithms (Binary, 3-nary, and 6-nary) as shown in Figure 2.

As shown in the Figure 2, lower split thresholds, such as 5% and 10%, consistently result in reduced average query counts for successful attacks. This is particularly evident in the 3-nary algorithm, which demonstrates the highest query efficiency across most thresholds. The finer-grained splits enabled by lower thresholds allow for more precise identification of impactful word substitutions, minimizing the number of queries required.

In contrast, higher split thresholds, such as 30% and 40%, generate larger segments that demand more exploration to identify optimal substitutions. While this approach captures more context, it significantly increases the query count, making it less efficient compared to finer splits.

Among the algorithms, the 6-nary algorithm exhibits slightly higher query counts at lower thresholds but maintains comparable attack accuracy. Meanwhile, the Binary Select consistently performs the least efficiently across all thresholds, requiring the highest query counts due to its limited segmentation and exploration capabilities.

C.3 Word Replacement Strategies

Alongside Wordnet replacement, we also explored a BERT focused replacement method by using mask-infill method where we mask the most influential word and find a replacement candidate (up to 5) using BERT. Table 12 (Appendix E) shows the results for N-Nary attack in IMDB using BERT replacement. From the results, there are no significant improvements when compared to the Wordnet replacement method.

C.4 The Impact of k on Query Count and Attack Success Rate (ASR)

We also evaluate how varying k (number of words allowed for replacement) affects query count and attack success rate (ASR) across standalone n-nary search methods (Binary, 3-Nary, 6-Nary, and Greedy) and the hybrid method with segmentation thresholds. Both K and segmentation thresholds present a trade-off between query efficiency and ASR. Figure 3 shows how ASR changes as k changes.

In both standalone and hybrid methods, smaller k values (e.g., $k = 3$) reduce query counts as fewer replacements are allowed, narrowing the search space. For instance, at $k = 3$, the 3-Nary method achieves the lowest query count. However, smaller k values limit ASR, as the restricted replacement budget fails to significantly degrade the model's accuracy. Across methods, ASR at $k = 3$ remains modest.

As k increases, query counts rise proportionally across all methods. At $k = 15$, ASR improves significantly as more words are replaced, allowing for more impactful attacks. The Greedy algorithm achieves the highest ASR at this configuration, though it requires more queries. Among n-nary methods, 3-Nary consistently strikes the best balance between query efficiency and ASR, while 6-Nary incurs the highest query counts due to its complex segmentation.

C.5 Dynamic-N Bins

Table 10 (Appendix D) contains the length bins and their corresponding Optimal-N values for IMDB, Yelp, and AGNews dataset. Based on the maximum length of the input text in validation dataset, 5 length bins are created and their corresponding optimal N values are obtained based on the query count. From the table, the n-value of *3* seems to be more optimal for most text length bins for the IMDB and Yelp datasets, *2* for AGNews. This also matches the results for hybrid and standalone methods, as the n-value of *3* has been shown to perform better in most cases.

D Length Bins and their Optimal-N

Length Bin	n
1 - 200	3
200 - 400	3
400 - 600	3
600 - 800	2
800 - max	6

Table 7: IMDB

Length Bin	n
1 - 135	3
135 - 270	3
270 - 405	3
405 - 540	2
540 - max	6

Table 8: Yelp

Length Bin	n
1 - 23	2
23 - 46	2
46 - 69	2
69 - 92	2
92 - max	2

Table 9: AGNews

Table 10: Length bins and their Optimal-N

E IMDB Tables

Table 11: Results Summary for N-nary Algorithm and Baseline(Binary and Greedy) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92.0	504	484	1.7	98
	Greedy	92.0	433	425	1.2	99
	3-Nary	92.0	475	461	1.5	98
	6-Nary	92.0	585	569	1.3	99
3	Binary	92.0	48	32	80.4	13
	Greedy	92.0	238	215	75.0	18
	3-Nary	92.0	46	35	81.1	12
	6-Nary	92.0	52	38	80.1	13
5	Binary	92.0	66	42	75.9	17
	Greedy	92.0	247	216	67.5	27
	3-Nary	92.0	64	45	74.3	19
	6-Nary	92.0	71	50	73.9	20
15	Binary	92.0	141	82	58.2	37
	Greedy	92.0	279	250	40.9	56
	3-Nary	92.0	133	83	54.5	41
	6-Nary	92.0	139	90	56.2	39

Table 12: Results Summary for N-Nary and Greedy Algorithms using BERT replacement stategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	504	483	1.8	98
	Greedy	92	432	422	1.4	99
	3-Nary	92	473	458	1.5	98
	6-Nary	92	582	570	1.3	99
3	Binary	92	47	31	80.4	13
	Greedy	92	238	214	75.0	18
	3-Nary	92	46	33	81.0	12
	6-Nary	92	52	37	80.1	13
5	Binary	92	66	41	75.9	17
	Greedy	92	247	215	67.5	26
	3-Nary	92	64	44	74.3	19
	6-Nary	92	71	50	73.9	20
15	Binary	92	141	82	58.2	37
	Greedy	92	279	249	40.9	55
	3-Nary	92	133	83	54.4	41
	6-Nary	92	138	89	56.2	39

Table 13: Results Summary for Hybrid Algorithm at Sentence Level (5% Split Threshold) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	375	363	998	1.8	98
	3-Nary	92	364	1053	1053	1.6	98
	6-Nary	92	392	380	974	1.9	98
3	Binary	92	51	37	53	79.2	14
	3-Nary	92	49	32	52	80.3	13
	6-Nary	92	50	34	52	79.9	13
5	Binary	92	66	44	71	74.6	17
	3-Nary	92	65	43	71	74.2	18
	6-Nary	92	66	43	71	74.8	17
15	Binary	92	126	78	157	55.6	40
	3-Nary	92	124	79	156	54.3	41
	6-Nary	92	126	83	156	54.2	41

Table 14: Results Summary for Hybrid Algorithm at Sentence Level (10% Split Threshold) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	378	368	943	1.7	98
	3-Nary	92	371	363	865	1.5	98
	6-Nary	92	387	374	945	2.1	98
3	Binary	92	54	38	56	79.9	14
	3-Nary	92	51	34	54	80.2	13
	6-Nary	92	50	34	52	79.6	14
5	Binary	92	70	46	75	75.1	17
	3-Nary	92	68	43	73	74.5	18
	6-Nary	92	66	44	71	74.3	18
15	Binary	92	129	81	160	55.5	40
	3-Nary	92	128	81	159	54.8	41
	6-Nary	92	125	81	156	54.1	41

Table 15: Results Summary for Hybrid N-Nary 5% Split Threshold with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	435	425	1251	1.2	99
	3-Nary	92	405	388	1363	1.6	98
	6-Nary	92	414	403	1200	1.3	98
3	Binary	92	49	35	51	80.3	13
	3-Nary	92	49	37	51	80.2	13
	6-Nary	92	50	35	52	80.5	13
5	Binary	92	65	43	70	75.5	17
	3-Nary	92	65	44	70	75.2	18
	6-Nary	92	65	43	70	75.0	18
15	Binary	92	127	80	155	57.5	38
	3-Nary	92	124	78	154	56.2	39
	6-Nary	92	123	79	152	55.3	40

Table 16: Results Summary for Hybrid N-Nary 10% Split Threshold with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	437	427	1117	1.3	99
	3-Nary	92	406	392	1320	1.4	98
	6-Nary	92	413	398	1327	1.5	98
3	Binary	92	57	38	60	80.9	12
	3-Nary	92	49	35	51	80.5	13
	6-Nary	92	49	33	52	80.5	13
5	Binary	92	73	47	78	75.3	18
	3-Nary	92	65	43	70	75.2	18
	6-Nary	92	65	43	70	74.9	18
15	Binary	92	133	83	163	58.4	37
	3-Nary	92	124	77	153	56.8	39
	6-Nary	92	123	79	152	55.3	40

Table 17: Results Summary for Hybrid N-Nary (20% Split Threshold) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	442	429	1526	1.1	99
	3-Nary	92	409	399	1279	1.1	99
	6-Nary	92	431	418	1302	1.3	98
3	Binary	92	74	49	77	80.6	13
	3-Nary	92	71	45	74	80.2	13
	6-Nary	92	88	66	92	79.1	14
5	Binary	92	89	56	97	75.0	18
	3-Nary	92	86	55	94	74.3	18
	6-Nary	92	106	75	114	72.8	21
15	Binary	92	145	91	177	57.1	38
	3-Nary	92	141	90	173	56.0	39
	6-Nary	92	160	114	192	54.2	41

Table 18: Results Summary for Hybrid N-Nary 30% Split Threshold with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	457	448	1136	1.2	99
	3-Nary	92	409	395	1430	1.2	99
	6-Nary	92	432	420	1302	1.3	98
3	Binary	92	102	73	107	79.8	13
	3-Nary	92	71	45	74	80.2	13
	6-Nary	92	88	66	92	79.1	14
5	Binary	92	117	82	125	74.1	19
	3-Nary	92	86	55	94	74.3	18
	6-Nary	92	106	75	114	72.8	21
15	Binary	92	168	117	204	53.8	42
	3-Nary	92	141	90	173	56.1	39
	6-Nary	92	160	114	192	54.3	41

Table 19: Results Summary for Hybrid N-Nary 40% Split Threshold with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Avg Query (For Fail)	Attack Accuracy (%)	ASR (%)
ALL	Binary	92	459	449	1167	1.3	99
	3-Nary	92	426	411	1324	1.5	98
	6-Nary	92	432	418	1479	1.2	99
3	Binary	92	102	72	107	79.8	13
	3-Nary	92	123	88	129	78.7	14
	6-Nary	92	88	66	92	79.1	14
5	Binary	92	117	82	125	74.0	19
	3-Nary	92	136	102	146	71.6	22
	6-Nary	92	106	75	114	72.7	21
15	Binary	92	168	117	204	53.9	42
	3-Nary	92	186	140	223	50.3	45
	6-Nary	92	160	114	192	54.4	41

Table 20: Results Summary for 5% Hybrid Threshold (Dyn-N + Hybrid) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)	ASR (%)
ALL	DynN+Hyb	92	408	390	1.6	98
3	DynN+Hyb	92	22	15	81.0	12
5	DynN+Hyb	92	35	23	74.5	19
15	DynN+Hyb	92	90	52	54.4	41

Table 21: Results Summary for 10% Hybrid Threshold (Dyn-N + Hybrid) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)	ASR (%)
ALL	DynN+Hyb	92	403	395	1.2	99
3	DynN+Hyb	92	22	16	80.9	12
5	DynN+Hyb	92	35	24	74.5	19
15	DynN+Hyb	92	90	52	54.5	41

Table 22: Results Summary for 20% Hybrid Threshold (Dyn-N + Hybrid) with WordNet Replacement Strategy

K (No. of Words)	Algorithm	Original Accuracy (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)	ASR (%)
ALL	DynN+Hyb	92	406	391	1.5	98
3	DynN+Hyb	92	22	16	81.0	12
5	DynN+Hyb	92	35	23	74.5	19
15	DynN+Hyb	92	90	52	54.6	41

Table 23: Results Summary for 30% Hybrid Threshold (Dyn-N + Hybrid) with WordNet Replacement Strategy

K (No. of Words)	**Algorithm**	**Original Accuracy (%)**	**Avg Query**	**Avg Query (For Atk Success)**	**Attack Accuracy (%)**	**ASR (%)**
ALL	DynN+Hyb	92	405	392	1.5	98
3	DynN+Hyb	92	22	15	80.9	12
5	DynN+Hyb	92	35	23	74.5	19
15	DynN+Hyb	92	90	51	54.5	41

Table 24: Results Summary for 40% Hybrid Threshold (Dyn-N + Hybrid) with WordNet Replacement Strategy

K (No. of Words)	**Algorithm**	**Original Accuracy (%)**	**Avg Query**	**Avg Query (For Atk Success)**	**Attack Accuracy (%)**	**ASR (%)**
ALL	DynN+Hyb	92	404	388	1.3	99
3	DynN+Hyb	92	22	16	80.9	12
5	DynN+Hyb	92	35	24	74.5	19
15	DynN+Hyb	92	90	52	54.6	41

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	92	90.5	471	456	1.5
3	Binary	92	11	47	33	81
5	Binary	92	17.5	65	45	74.5
15	Binary	92	37.6	133	83	54.4

Table 25: Results Summary for Dynamic-N

F Yelp Tables

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	97	98	401	400	2
	Greedy	97	98	327	329	2
	3-Nary	97	98	375	374	2
	6-Nary	97	98	448	449	2
3	Binary	97	11	46	32	86
	Greedy	97	13	155	103	84
	3-Nary	97	11	43	32	86
	6-Nary	97	11	49	35	86
5	Binary	97	17	66	43	80
	Greedy	97	21	166	106	76
	3-Nary	97	18	61	42	79
	6-Nary	97	17	69	46	80
15	Binary	97	41	139	87	57
	Greedy	97	52	207	155	45
	3-Nary	97	42	133	86	55
	6-Nary	97	39	138	88	58

Table 26: Performance results for N-Nary in Yelp

K (No. of words)	**Replacement Strategy**	**Algorithm**	**Original Accuracy (%)**	**ASR (%)**	**Avg Query**	**Avg Query (For Atk Success)**	**Attack Accuracy (%)**
ALL	BERT (top_k = 5)	Binary	97	94.6	403	402	2.4
		Greedy	97	94.9	326	328	2.1
		3-Nary	97	94.6	375	374	2.4
		6-Nary	97	95.3	449	449	1.7
3	BERT (top_k = 5)	Binary	97	11	46	33	86
		Greedy	97	13.1	155	103	83.9
		3-Nary	97	11.4	43	32	85.6
		6-Nary	97	11.2	50	36	85.8
5	BERT (top_k = 5)	Binary	97	16.9	66	44	80.1
		Greedy	97	21.4	166	106	75.6
		3-Nary	97	17.6	61	41	79.4
		6-Nary	97	16.6	69	47	80.4
15	BERT (top_k = 5)	Binary	97	40.5	139	88	56.5
		Greedy	97	52.1	207	155	44.9
		3-Nary	97	42.4	132	85	54.6
		6-Nary	97	39.1	138	89	57.9

Table 27: Performance results for BERT replacement strategy in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Dyn-N	97	98	379	378	2
3	Dyn-N	97	11	44	33	86
5	Dyn-N	97	18	62	43	79
15	Dyn-N	97	42	133	86	55

Table 28: Performance results for Dynamic-N in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	97	98	366	368	305	2
	3-Nary	97	98	347	345	414	2
	6-Nary	97	98	335	332	475	2
3	Binary	97	11	47	33	49	86
	3-Nary	97	12	45	32	47	85
	6-Nary	97	12	47	33	49	86
5	Binary	97	17	64	43	69	80
	3-Nary	97	17	62	42	66	80
	6-Nary	97	17	63	42	67	80
15	Binary	97	40	130	85	161	57
	3-Nary	97	39	127	81	158	58
	6-Nary	97	42	126	83	158	56

Table 29: Performance results for N-Nary in Hybrid(5%) in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	97	97	364	363	410	2
	3-Nary	97	97	347	346	404	2
	6-Nary	97	97	334	331	475	2
3	Binary	97	11	50	34	53	86
	3-Nary	97	11	46	33	47	86
	6-Nary	97	12	47	34	48	85
5	Binary	97	17	68	41	73	81
	3-Nary	97	17	62	42	66	80
	6-Nary	97	17	63	42	67	80
15	Binary	97	38	131	83	162	59
	3-Nary	97	39	125	79	157	58
	6-Nary	97	42	125	83	157	55

Table 30: Performance results for N-Nary in Hybrid(10%) in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	97	95	366	365	413	2
	3-Nary	97	95	348	349	268	2
	6-Nary	97	95	336	333	472	2
3	Binary	97	12	61	40	64	85
	3-Nary	97	11	58	37	61	86
	6-Nary	97	12	70	40	74	85
5	Binary	97	17	79	44	87	80
	3-Nary	97	18	75	45	81	80
	6-Nary	97	18	88	52	96	79
15	Binary	97	37	139	81	174	60
	3-Nary	97	37	136	81	170	60
	6-Nary	97	39	147	87	187	58

Table 31: Performance results for N-Nary in Hybrid(20%) in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	97	95	368	366	465	2
	3-Nary	97	95	347	349	267	2
	6-Nary	97	95	336	333	427	2
3	Binary	97	12	82	46	87	85
	3-Nary	97	11	58	37	61	86
	6-Nary	97	12	70	40	74	85
5	Binary	97	18	98	53	109	79
	3-Nary	97	18	75	45	81	80
	6-Nary	97	18	88	52	96	79
15	Binary	97	39	154	92	197	58
	3-Nary	97	37	136	81	170	60
	6-Nary	97	39	147	87	187	58

Table 32: Performance results for N-Nary in Hybrid(30%) in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	97	95	367	366	418	2
	3-Nary	97	95	356	354	400	2
	6-Nary	97	95	337	335	441	2
3	Binary	97	12	82	46	87	85
	3-Nary	97	12	95	53	101	85
	6-Nary	97	13	70	41	74	84
5	Binary	97	18	98	53	109	79
	3-Nary	97	19	110	62	122	78
	6-Nary	97	18	88	51	96	79
15	Binary	97	39	154	92	197	58
	3-Nary	97	41	165	102	213	56
	6-Nary	97	39	147	87	187	58

Table 33: Performance results for N-Nary in Hybrid(40%) in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	97	95	321	320	2
3	DynN+Hyb	97	12	22	17	86
5	DynN+Hyb	97	18	35	25	79
15	DynN+Hyb	97	42	95	60	55

Table 34: Performance results for Dynamic-N+Hybrid with Split Threshold 5% in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	97	95	321	319	2
3	DynN+Hyb	97	12	22	18	86
5	DynN+Hyb	97	18	34	25	79
15	DynN+Hyb	97	42	95	60	55

Table 35: Performance results for Dynamic-N+Hybrid with Split Threshold 10% in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	97	95	321	319	2
3	DynN+Hyb	97	12	22	17	86
5	DynN+Hyb	97	18	34	25	79
15	DynN+Hyb	97	42	95	59	55

Table 36: Performance results for Dynamic-N+Hybrid with Split Threshold 20% in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	97	95	321	319	2
3	DynN+Hyb	97	12	22	18	86
5	DynN+Hyb	97	18	35	25	79
15	DynN+Hyb	97	42	95	60	55

Table 37: Performance results for Dynamic-N+Hybrid with Split Threshold 30% in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	96.91	94.9	323	321	2
3	DynN+Hyb	97	11.4	22	17	86
5	DynN+Hyb	97	17.6	34	25	79
15	DynN+Hyb	96.91	41.8	94	59	55

Table 38: Performance results for Dynamic-N+Hybrid with Split Threshold 40% in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	97	94	328	327	3
	3-Nary	97	93.6	321	316	3.4
	6-Nary	97	94.2	337	332	2.8
3	Binary	97	11.4	46	31	85.6
	3-Nary	97	11.3	45	29	85.7
	6-Nary	97	11.3	47	34	85.7
5	Binary	97	17.3	64	41	79.7
	3-Nary	97	17.4	62	42	79.6
	6-Nary	97	17.1	65	44	79.9
15	Binary	97	40	127	83	57
	3-Nary	97	40.4	126	81	56.6
	6-Nary	97	39.9	130	86	57.1

Table 39: Performance results for Hybrid algorithm(5%) at Sentence Level in Yelp

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	97	94	328	327	3
	3-Nary	97	94.2	321	320	2.8
	6-Nary	97	93.9	331	325	3.1
3	Binary	97	11	47	30	86
	3-Nary	97	11.7	46	30	85.3
	6-Nary	97	11.4	46	34	85.6
5	Binary	97	17.7	65	42	79.3
	3-Nary	97	17.9	62	43	79.1
	6-Nary	97	17.6	63	44	79.4
15	Binary	97	39.3	127	82	57.7
	3-Nary	97	40.1	124	78	56.9
	6-Nary	97	39.7	127	84	57.3

Table 40: Performance results for Hybrid algorithm(10%) at Sentence Level in Yelp

G AGNews Tables

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	73	185	154	25
	Greedy	95	73	185	154	25
	3-Nary	95	72	202	162	26
	6-Nary	95	76	304	253	23
3	Binary	95	7	50	44	88
	Greedy	95	7	50	44	88
	3-Nary	95	7	33	29	88
	6-Nary	95	6	37	29	89
5	Binary	95	12	61	53	83
	Greedy	95	12	61	53	83
	3-Nary	95	11	49	34	85
	6-Nary	95	10	53	39	85
15	Binary	95	35	117	89	60
	Greedy	95	35	117	89	60
	3-Nary	95	31	118	81	64
	6-Nary	95	26	116	77	69

Table 41: Performance results for N-Nary in AGNews

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	73	178	140	26
	3-Nary	95	72	178	139	26
	6-Nary	95	77	275	220	23
3	Binary	95	6	18	16	89
	3-Nary	95	7	18	16	88
	6-Nary	95	6	16	12	89
5	Binary	95	9	32	26	86
	3-Nary	95	11	31	20	85
	6-Nary	95	10	28	20	85
15	Binary	95	32	96	63	63
	3-Nary	95	33	96	62	64
	6-Nary	95	28	89	56	68

Table 42: Performance results for AGNews with Hybrid 5%

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	72	178	140	27
	3-Nary	95	72	178	138	27
	6-Nary	95	76	277	221	23
3	Binary	95	6	18	15	89
	3-Nary	95	7	18	16	88
	6-Nary	95	6	16	12	89
5	Binary	95	9	32	25	86
	3-Nary	95	11	31	21	85
	6-Nary	95	11	28	20	85
15	Binary	95	32	96	63	63
	3-Nary	95	33	96	62	64
	6-Nary	95	28	89	55	69

Table 43: Performance Metrics for AGNews with Hybrid 10%

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	74	178	141	26
	3-Nary	95	72	178	138	27
	6-Nary	95	76	278	224	23
3	Binary	95	6	18	16	89
	3-Nary	95	7	18	16	88
	6-Nary	95	6	16	13	89
5	Binary	95	9	32	27	86
	3-Nary	95	11	31	21	85
	6-Nary	95	11	28	21	85
15	Binary	95	33	97	64	63
	3-Nary	95	33	96	63	64
	6-Nary	95	28	89	56	69

Table 44: Performance results for AGNews with Hybrid 20%

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	74	179	141	27
	3-Nary	95	72	179	139	27
	6-Nary	95	76	277	220	23
3	Binary	95	6	18	16	89
	3-Nary	95	7	18	16	88
	6-Nary	95	6	16	12	89
5	Binary	95	9	32	26	86
	3-Nary	95	11	31	21	85
	6-Nary	95	10	28	20	85
15	Binary	95	33	97	64	63
	3-Nary	95	33	96	63	64
	6-Nary	95	28	89	56	69

Table 45: Performance results for AGNews with Hybrid 30%

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Binary	95	74	179	141	27
	3-Nary	95	73	178	140	26
	6-Nary	95	76	277	218	23
3	Binary	95	6	18	15	89
	3-Nary	95	7	18	17	88
	6-Nary	95	6	16	12	89
5	Binary	95	9	32	26	86
	3-Nary	95	11	31	22	85
	6-Nary	95	10	28	20	85
15	Binary	95	33	97	64	64
	3-Nary	95	33	96	63	64
	6-Nary	95	28	89	56	69

Table 46: Performance results for AGNews with Hybrid 40%

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Dyn N	95	74	209	169	27
3	Dyn N	95	7	33	28	88
5	Dyn N	95	11	49	35	85
15	Dyn N	95	26	116	77	69

Table 47: Performance results for AGNews with Dynamic-N

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Dyn N + Hyb	95	72	179	141	27
3	Dyn N + Hyb	95	7	18	16	88
5	Dyn N + Hyb	95	11	31	21	85
15	Dyn N + Hyb	95	26	89	55	69

Table 48: Performance results for AGNews with Dynamic N + Hybrid(5%) algorithm

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	Dyn N + Hyb	95	73	179	141	27
3	Dyn N + Hyb	95	7	18	16	88
5	Dyn N + Hyb	95	10	31	21	85
15	Dyn N + Hyb	95	26	89	56	69

Table 49: Performance results for AGNews with Dynamic N + Hybrid(10%) algorithm

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	95	73	179	140	27
3	DynN+Hyb	95	7	18	17	88
5	DynN+Hyb	95	10	31	22	85
15	DynN+Hyb	95	26	89	55	69

Table 50: Performance results for AGNews with Dynamic N + Hybrid(20%) algorithm

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	95	72	178	141	26
3	DynN+Hyb	95	7	18	16	88
5	DynN+Hyb	95	10	31	20	85
15	DynN+Hyb	95	26	89	55	69

Table 51: Performance results for AGNews with Dynamic N + Hybrid(30%) algorithm

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Attack Accuracy (%)
ALL	DynN+Hyb	95	72	178	140	27
3	DynN+Hyb	95	7	18	16	88
5	DynN+Hyb	95	10	31	21	85
15	DynN+Hyb	95	26	89	55	69

Table 52: Performance results for AGNews with Dynamic N + Hybrid(40%) algorithm

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	95	72	200	160	304	26
	3-Nary	95	70	201	164	303	25
	6-Nary	95	73	218	172	338	26
3	Binary	95	6	34	25	34	89
	3-Nary	95	7	34	29	34	88
	6-Nary	95	6	35	27	35	89
5	Binary	95	9	50	37	51	86
	3-Nary	95	11	49	36	50	84
	6-Nary	95	10	52	38	53	85
15	Binary	95	33	117	84	135	62
	3-Nary	95	30	117	80	134	65
	6-Nary	95	30	119	81	137	65

Table 53: Performance results for AGNews with Sentence level + Hybrid(10%)

K (No. of words)	Algorithm	Original Accuracy (%)	ASR (%)	Avg Query	Avg Query (For Atk Success)	Avg Query for fail	Attack Accuracy (%)
ALL	Binary	95	73	191	153	292	26
	3-Nary	95	73	191	153	291	26
	6-Nary	95	73	189	154	285	26
3	Binary	95	6	34	24	35	89
	3-Nary	95	7	34	27	34	88
	6-Nary	95	6	33	25	34	89
5	Binary	95	10	49	34	50	85
	3-Nary	95	10	48	35	49	85
	6-Nary	95	10	49	34	51	85
15	Binary	95	32	112	80	128	63
	3-Nary	95	32	111	80	126	63
	6-Nary	95	32	112	79	130	63

Table 54: Performance results for AGNews with Sentence level + Hybrid(20%)