Indian Statistical Institute

Master's Thesis

Exploring Character-level Attacks on Neural Ranking Models

Author:
Surjyanee Halder

Supervisor:
Dr. Debapriyo Majumdar

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology

in

Computer Science

June 16, 2025

Indian Statistical Institute
203, B.T. Road, Kolkata : 700108

CERTIFICATE

I certify that I have read the thesis titled **Exploring Character-level Attacks on Neural Ranking Models**, prepared under my guidance by **Surjyanee Halder**, and in my opinion it is fully adequate, in quality and in scope, as a dissertation for the degree of **Master of Technology in Computer Science** of the Indian Statistical Institute.

Debapriyo Majumdar
Assistant Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute

Kolkata
June, 2025.

Declaration of Authorship

I, Surjyanee Halder, declare that this thesis titled, "Exploring Character-level Attacks on Neural Ranking Models" and the work presented in it are my own. I confirm that:

- This work was done mainly while in candidature for a Master's degree at ISI Kolkata.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, I have given the source. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all the sources of help.

Signed:

_____

Date:

_____

INDIAN STATISTICAL INSTITUTE

Abstract

Exploring Character-level Attacks on Neural Ranking Models

Neural ranking models (NRMs) have achieved state-of-the-art performance in information retrieval, yet they remain highly susceptible to subtle adversarial inputs such as character-level typos. This project explores the robustness of such systems by introducing a reinforcement learning (RL)-based query perturbation framework. RL agents—PPO, DQN, and A2C—were trained to minimally modify user queries (e.g., through character deletions or swaps) with the goal of significantly altering the resulting document rankings, as measured by Kendall's Tau. Experiments were conducted on the TREC DL 2019 and 2020 benchmarks using two different neural rankers: MiniLM and a fine-tuned CharacterBERT model. The perturbation attacks were shown to succeed in over 85% of cases for MiniLM and approximately 40% for CharacterBERT, indicating varying degrees of vulnerability. To mitigate these effects, a set of pretrained query recovery models—such as T5-large-spell, spelling-correction-base, and grammar correction modules—were applied to restore the original query form. When used in combination, these recovery mechanisms reduced the MiniLM attack success rate to around 52%, demonstrating partial robustness. This study underscores both the fragility of neural rankers to character-level noise and the value of lightweight correction pipelines in improving retrieval resilience.

Acknowledgements

I would like to express my sincere gratitude to my research guides, Dr. Mandar Mitra and Dr. Debapriyo Majumdar, for their invaluable guidance and support throughout this research. Their expertise and insights have been instrumental in shaping this thesis.

Special thanks to Sourav Saha, research scholar pursuing his Ph.D, who guided me through various aspects of this research. His patience and knowledge were crucial in overcoming numerous challenges.

I would like to thank my classmate, Tanmay Karmakar, for his collaboration. His support and cooperation have made this journey more manageable and enjoyable.

Thank you all for your patient support and belief in my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural Ranking Models (NRMs) have emerged as a powerful paradigm in information retrieval, driven by the rise of deep learning and the need to capture complex semantic relationships between queries and documents. Traditional retrieval methods like BM25 relied heavily on exact keyword matching, often failing to understand contextual meaning or handle diverse query formulations. NRMs address this by learning dense vector representations that encode rich semantic features, enabling more accurate and relevant document ranking. Initially inspired by advances in representation learning and language modeling (e.g., BERT, transformers), these models evolved to incorporate architectures specifically tuned for ranking tasks-ranging from interaction-focused models to representation-based frameworks. Their integration into reinforcement learning (RL) pipelines allows for dynamic environments where retrieval quality can be improved through reward-driven optimization. Despite their performance, NRMs remain vulnerable to small input perturbations, underscoring the need to evaluate and enhance their robustness in adversarial and noisy settings.

1.1 Problem of Robustness

Despite their impressive capabilities, NRMs are not robust to small input perturbations, especially in the form of typos or adversarial edits. A seemingly trivial typo like “definiiton” instead of “definition” can significantly alter the model’s internal representation of the query, leading to poor retrieval results or even irrelevant outputs.

This lack of robustness is especially concerning in high-stakes applications such as legal search, clinical question-answering, and digital assistants. If minor, human-like errors can drastically shift rankings, then the reliability of these systems is compromised.

1.2 Adversarial Attacks on IR Systems

Adversarial attacks are deliberate modifications to input queries or documents designed to manipulate the output of a model. While much work has been done on adversarial attacks in computer vision and natural language understanding, attacks on neural IR systems are less explored but increasingly important.

In IR, such attacks often involve minimal edits to a query-like adding, deleting, or substituting characters-intended to disrupt the ranking order produced by the system. These are known as character-level attacks. What makes these attacks powerful is their subtlety: the perturbed queries often appear natural or human-typed, yet they can significantly degrade retrieval performance.

Adversarial attacks on NRMs can be categorized based on Attacker's Knowledge. In **White-Box Attacks** the attacker is granted the full access and knowledge of the target model, including its architecture, parameters (weights and biases), and potentially the training data. This allows for more precise and often more effective attacks, often using gradient information to craft perturbations. While in **Gray-Box Attacks** the attacker has partial knowledge about the target model, perhaps knowing the architecture but not the specific weights, or having access to confidence scores but not gradients. But in **Black-Box Attacks** the attacker has no knowledge about internal workings of the model and can only interact with it by giving inputs and observing the outputs (e.g., class labels or confidence scores) [14]. These are more realistic in many practical scenarios. Attack strategies often involve querying the model multiple times to infer vulnerabilities or training a local substitute model.

1.3 Problem Statement

Neural ranking models (NRMs), such as MiniLM and CharacterBERT, have demonstrated strong performance on various information retrieval (IR) tasks. However, these models are vulnerable to small, adversarial perturbations in user queries-especially at the character level, such as typographical errors. These seemingly benign modifications can significantly alter the retrieved document rankings, thus impacting both user satisfaction and system reliability.

The goal is to minimally modify a given query such that the resulting ranked list of documents changes significantly.

Formally, given
a query q and a document ranking function $f(\cdot)$, our objective is to find a minimally perturbed query $\tilde{q}$ such that:

$$\tau(f(q), f(\tilde{q})) < \delta, \tag{1.1}$$

where $\tau(\cdot, \cdot)$ is the Kendall's Tau rank correlation, and δ is a threshold indicating a significant ranking shift (e.g., $\delta = 0.5$). The perturbation should be minimal:

$$\text{EditDistance}(q, \tilde{q}) \leq \epsilon, \tag{1.2}$$

with ϵ being a small integer (typically 1 or 2) to mimic realistic typos.

We will also explore query recovery by training a model $g(\cdot)$ such that:

$$\hat{q} = g(\tilde{q}) \quad \text{and} \quad \tau(f(q), f(\hat{q})) \geq \gamma, \tag{1.3}$$

where $\hat{q}$ is the recovered query and γ is a threshold indicating successful recovery.

1.4 Motivation

Understanding how and why neural IR systems fail under minor perturbations is essential for building robust, trustworthy models. As search engines become deeply integrated into our daily lives-from casual browsing to academic research and enterprise knowledge discovery-their robustness becomes a matter of usability, accessibility, and fairness.

Moreover, our two-part system-(1) generating typo-based perturbations and (2) reconstructing the original queries-offers a testbed for evaluating and improving robustness in future IR models. By simulating human-like mistakes and gauging recovery performance, we can both test and strengthen existing ranking architectures.

Chapter 2

Related Works

The vulnerability of deep neural networks to adversarial examples, which are inputs with small, often imperceptible perturbations designed to cause incorrect model outputs, was first demonstrated in the image domain in [13]. This discovery has since spurred a significant amount of research into the robustness of models in other domains, including Natural Language Processing (NLP). For Neural Machine Translation (NMT), which relies on complex sequence-to-sequence models, understanding these vulnerabilities is critical for building reliable systems. This review categorizes and discusses adversarial attacks on NMT systems based on the granularity of the input perturbation: character, word, and sentence level.

Character-Level Attacks

Character-level attacks manipulate the source text by introducing subtle errors such as insertions, deletions, swaps, or visual substitutions of characters. These modifications are often difficult for a human reader to notice but can cause a catastrophic failure in the translation output. The primary goal is to break the model's processing of sub-word units or its reliance on correct orthography while maintaining the visual and semantic integrity of the source text for a human observer.

Ebrahimi [6] was among the first to systematically explore character-level attacks against NMT. They proposed a gradient-based method to identify characters that, when manipulated, would maximally disrupt the translation. Their "targeted attack" could force the model to produce a specific, malicious phrase in the output, while "untargeted attacks" simply aimed to degrade the translation quality, as measured by BLEU score. Another common technique involves leveraging visual similarity between characters (e.g., replacing '0' with 'O' or

'l' with 'I') to create adversarial examples that are nearly identical to the original input [7]. These attacks highlight the brittleness of models that rely on character-level tokenization or have not been explicitly trained to be robust against such typographic noise.

Word-Level Attacks

Word-level attacks operate by replacing one or more words in the source sentence with alternatives. For the attack to be successful, the replacement word must be contextually appropriate to avoid easy detection, yet different enough to fool the model. Early approaches focused on replacing words with their synonyms, typically sourced from lexical databases like WordNet, by Ribeiro, Singh, and Guestrin [11]. While effective, this method is limited by the coverage of the database and its inability to account for subtle contextual nuances.

More advanced techniques leverage the distributional representations learned by neural networks. Cheng [4] proposed a method that replaces a source word with another word that is close in the embedding space but causes a significant change in the translation output. This approach is more flexible than synonym replacement but still risks generating semantically incongruous sentences. To address this, recent work has focused on preserving semantic consistency. Garg [8] proposed methods that use pre-trained masked language models (e.g., BERT) to find plausible, context-aware substitutes. By masking a word and having the language model predict high-probability replacements, they ensure that the adversarial sentence remains fluent and semantically coherent, making the attack far more potent and stealthy. These attacks demonstrate that even when preserving local semantics, the global compositional meaning understood by the NMT model can be easily manipulated.

Sentence-Level (or Phrase-Level) Attacks

These Sentence-level attacks involve more substantial modifications to the source input, such as paraphrasing, changing the syntactic structure, or appending distracting phrases. These attacks are semantically meaning-preserving from a human perspective but exploit the model's sensitivity to surface-level form and syntax.

One prominent technique is back-translation, where a sentence is translated to a pivot language and then translated back to the original language to generate a diverse set of paraphrases [10]. By selecting a paraphrase that degrades the target translation, an adversary can create a valid adversarial example.

Another powerful method involves applying syntactically controlled transformations. Iyyer also demonstrated the use of rules to change the syntactic structure of a sentence (e.g., from active to passive voice) without altering its semantic content, leading to significant drops in NMT performance.

Furthermore, research has shown that NMT models are susceptible to "distractor" or "padding" attacks. Belinkov [1] found that simply appending a short, syntactically valid but semantically unrelated phrase (e.g., "and that is it") to the end of a source sentence could severely damage the quality of the translation. This suggests that the attention mechanisms in NMT models can be easily distracted, failing to correctly weigh the significance of different parts of the source sentence and exposing a fundamental weakness in how they model long-range dependencies.

2.1 Typo Queries

Handling typographical errors (typos) in user queries is a foundational challenge in IR, representing a natural form of character-level perturbation [9]. Historically, this was addressed using dictionary-based methods and edit distance metrics [3], but the state-of-the-art has since shifted to neural sequence-to-sequence models. Modern approaches, particularly Transformer architectures like T5, leverage large-scale, noisy-to-clean text corpora to perform robust grammatical and spelling correction [12].

While robustness to typos is a crucial first step, it is distinct from robustness against malicious, character-level adversarial attacks. Character-aware models such as CharacterBERT have demonstrated strong performance on both tasks, suggesting a shared underlying requirement [2]. Nevertheless, a key distinction remains: adversarial examples are intentionally crafted to be worst-case perturbations designed to fool models and bypass standard correctors, unlike the more random and predictable nature of common typos [7]. This necessitates dedicated research into adversarial defense beyond conventional typo-handling techniques.

2.2 BERT (Subword) Tokenizers vs. Character-Based Tokenizers

The choice of the tokenizer has significant impacts on how a model "sees" text and its inherent robustness to character-level variations.

2.2.1 BERT (Subword) Tokenizers (e.g., WordPiece, Byte Pair Encoding - BPE)

- Mechanism: These tokenizers break words into common or frequently occurring sub-word units. For example, "tokenization" might become "token" and "##ization". This helps manage vocabulary size, handle rare words (by breaking them into known subwords), and share representations among related words.
- Fragility: A single character change in a word can lead to a completely different subword segmentation. For instance, "important" might be one token, but "impotrant" (typo) could be split into "imp", "##tran" or become an <UNK> (unknown) token if the subword parts are not in the vocabulary or are rare. This dramatically changes the input representation for the model.
- "Curse of Tokenization": Research highlights that subword tokenizers can be sensitive to typos, length variations, and often ignore the internal structure or compositionality of tokens.
- Out-of-Vocabulary (OOV) for perturbed subwords: Even if a word is known, a slight misspelling might make its subwords unknown or rare, leading to poor embeddings.

2.2.2 Character-Based Tokenizers

- Mechanism: Each character is treated as a distinct token. The vocabulary consists of all possible characters in the language(s) being handled.
- Inherent Typo Tolerance: Typos are just different character sequences; they don't create OOV tokens in the same way subword tokenizers might produce OOV subwords. The model learns representations for individual characters and their combinations.
- Handling OOV Words: New or rare words are naturally handled as sequences of known characters.
- Fine-grained Analysis: Useful for tasks sensitive to spelling or requiring granular analysis.

- Improved Robustness to Adversarial Attacks: Several studies suggest that models using character-level inputs or character embeddings are more robust to character-level adversarial attacks and spelling errors. For example, ByT5 [15] (a byte-level T5 model), CharacterBERT [2] and CANINE [5] (character-level model) are designed with this principle.

Token-free models like ByT5 and CANINE achieve robustness by completely removing the traditional subword vocabulary. ByT5 operates directly on raw UTF-8 bytes, making it inherently immune to out-of-vocabulary errors and orthographic tricks like homoglyphs, as visually similar characters have different byte representations. Similarly, CANINE processes text as a sequence of Unicode characters but cleverly uses a downsampling mechanism to summarize long character sequences into a manageable length for its Transformer, making it highly efficient. Both models avoid the pitfalls of a fixed tokenizer, allowing them to handle any text, language, or noise without failure.

In contrast, CharacterBERT uses a hybrid approach that enriches word-level representations with character-level understanding. It first tokenizes text into words and then uses a Character-level Convolutional Neural Network (CharCNN) to analyze the character composition of each word. This process generates a robust, character-aware embedding for every word, which is then fed into a standard BERT architecture. This method allows the model to handle typos and unknown words gracefully, as it can infer a word's meaning from its characters rather than mapping it to a generic "unknown" token, thus preserving vital information.

In summary, prior work has extensively explored both subword and character-level tokenization strategies to improve robustness against adversarial attacks and handling of OOV words. While subword models like BERT have shown strong performance, they remain vulnerable to small perturbations due to their reliance on token boundaries. On the other hand, character-level models demonstrate enhanced resilience to typographical errors and adversarial noise.

These insights motivate our exploration of robustness in IR systems through character-level perturbations and recovery techniques.

Chapter 3

Proposed Methods

This section outlines the two-stage framework developed to analyze and mitigate the vulnerability of Neural Ranking Models (NRMs) to adversarial character-level perturbations. The overall pipeline comprises: (1) a reinforcement learning agent that strategically introduces minimal yet impactful query perturbations to degrade ranking quality, and (2) a suite of fine-tuned T5-based models designed to recover the original query from its perturbed version, thereby reducing adversarial effectiveness.

3.1 Model 1: RL-Based Query Perturbation

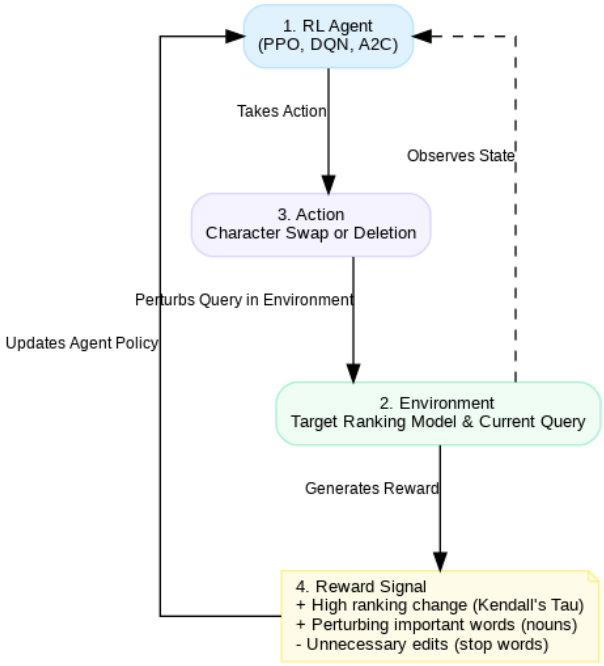

FIGURE 3.1: Framework for Model-1

The first model employs reinforcement learning (RL) to learn optimal character-level perturbation strategies that can adversarially affect a neural ranker's performance. Given a query and a black-box access to a target ranking model (such as MiniLM Cross-Encoder or CharacterBERT), the RL agent perturbs the query by performing one of a limited set of operations at each timestep.

The choice to constrain the action space was a deliberate design decision aimed at generating realistic, human-like typos. While a broader set of four character-level edits—insertion, deletion, substitution, and swap (transposition)—was initially considered, we opted to focus on **deletions** and **swaps**. This is because random insertions and substitutions can often produce less plausible word forms, whereas accidental single-character deletions and adjacent character swaps are highly frequent in genuine user queries. By restricting the action space, we guide the RL agent to learn the most impactful yet minimally disruptive perturbations, ensuring the adversarial examples remain challenging while closely mimicking natural user behavior.

Algorithm 1: Training of RL-agents for Minimal Query Perturbation

Require: Query set Q, Document set D, Pretrained Ranker $\mathcal{R}$, RL Algorithm $\mathcal{A}$, Timesteps T

Ensure: Trained RL agent π^*

Initialize RL Environment $\mathcal{E}$ with queries Q and documents D
Initialize RL Agent π using $\mathcal{A}$ (e.g., DQN, PPO, A2C)
for $t = 1$ to T **do**
    Sample query $q \sim Q$
    Reset environment $\mathcal{E}$ with q
    for each step until max steps **do**
        Observe token-level features o_t from q
        Choose action $a_t \sim \pi(o_t)$ ▷ Delete or Transpose character
        Apply a_t to get perturbed query $\tilde{q}$
        Compute ranking lists: $r_q \leftarrow \mathcal{R}(q, D_q)$, $r_{\tilde{q}} \leftarrow \mathcal{R}(\tilde{q}, D_q)$
        Compute Kendall's τ shift $\Delta\tau = 1 - \tau(r_q, r_{\tilde{q}})$
        Compute edit distance penalty λ
        Compute reward:

$$R_t = 10 \cdot \Delta\tau - \lambda$$

        Store (o_t, a_t, R_t, o_{t+1}) in experience buffer
    end for
    Update policy π using stored experience
end for
return Trained agent π^*

The agent is trained using popular RL algorithms such as PPO, DQN, and A2C. The state consists of the current character-level query string (optionally encoded with token embeddings), while the action space allows indexed character operations. The reward function is carefully designed to encourage high-ranking list changes (measured by Kendall's Tau shift) while penalizing unnecessary edits or edits on unimportant tokens (e.g., stop words). Additional positive rewards are introduced for perturbing content-bearing parts of the query, such as named entities or nouns. Training proceeds over multiple episodes per query, aiming to learn perturbations that can degrade the ranker's output significantly, even with minor modifications.

Algorithm 2: Reinforcement Learning Framework for Adversarial Query Generation

Input: Query set Q, document sets D, ranking model M_{rank}, RL agents $\Pi = \{\pi_{\text{PPO}}, \pi_{\text{DQN}}, \pi_{\text{A2C}}\}$.
Output: Comparative analysis of agent performance in generating adversarial queries.

procedure SETUP
    Load TREC DL dataset for query-document pairs (Q, D).
    Load pre-trained cross-encoder model M_{rank}.
    Define RL environment 'QueryPerturbationEnv' with:
        State s_t: Feature matrix of the current query (token IDs, POS tags, etc.).
        Action a_t: Discrete action {Delete Char, Transpose Char}.
        Reward r_t: $R = (1 - \tau) \times 20$, where τ is Kendall's Tau.
end procedure

procedure EVALUATEAGENTS
    Initialize results dictionary 'AllMetrics'.
    for each agent π_{agent} in Π **do**
        Initialize storage for rewards, τ-shifts, and edit distances.
        for each query $q_i \in Q$ over fixed episodes **do**
            Reset environment with original query q_i.
            done $\leftarrow$ false.
            while not *done* **do**
                Observe state s_t.
                Select action $a_t \leftarrow \pi_{\text{agent}}(s_t)$.
                Execute a_t to get q', s_{t+1}, r_t, and *done* status.
                // Env calculates r_t by ranking D_i with q_i and q' using M_{rank}.
                Store metrics (r_t, τ-shift, edit distance).
            end while
        end for
        Aggregate metrics to compute avg τ-shift, avg edit distance, and success rate.
        Store aggregated results in 'AllMetrics'.
    end for
    return 'AllMetrics'
end procedure

3.2 Model 2: T5-Based Query Recovery

The second model attempts to reverse the adversarial effect introduced by Model 1 by recovering the original query from its perturbed form. For this, a set of pre-trained and fine-tuned T5 variants are employed, each designed to perform some form of spelling or grammatical correction.

Algorithm 3: Evaluating Few-Shot Performance of a T5 Correction Model

1: **Input:** A T5 model M, its tokenizer T, a dataset D of (perturbed, original) text pairs.
2: **Parameters:** Number of prompt examples k. Note: if $k = 0$, the evaluation is zero-shot.
3: **Initialize:** Load model M and tokenizer T.
4: Split dataset D into an example pool D_{ex} and a test set D_{test}.
5: Set correct_predictions $\leftarrow 0$.
6: **for** each pair $(q_{\text{pert}}, q_{\text{orig}})$ in D_{test} **do**
7: Construct a prompt P for q_{pert}, optionally including k examples from D_{ex}.
8: Generate a predicted correction: $q_{\text{pred}} \leftarrow M.\text{generate}(T(P))$.
9: **if** q_{pred} exactly matches q_{orig} **then**
10: correct_predictions $\leftarrow$ correct_predictions + 1
11: **end if**
12: **end for**
13: **Calculate Accuracy:** Accuracy $\leftarrow$ (correct_predictions/$|\text{D}_{\text{test}}|$) $\times$ 100.
14: **Output:** The final prediction accuracy of the model.

These models are fine-tuned (where applicable) on noisy-to-clean spelling datasets and evaluated on their ability to restore the original query such that the resulting ranking (using the same NRM) aligns closely with the one generated by the clean query. Success is measured by the recovery model's ability to reduce the adversarial impact (i.e., improving Kendall's Tau between recovered and original rankings). When integrated with the RL model in a pipeline, these recovery models were shown to reduce the attack success rate.

Together, these two models form a robust adversarial testing and recovery system for neural rankers, enabling both targeted attack simulation and defense

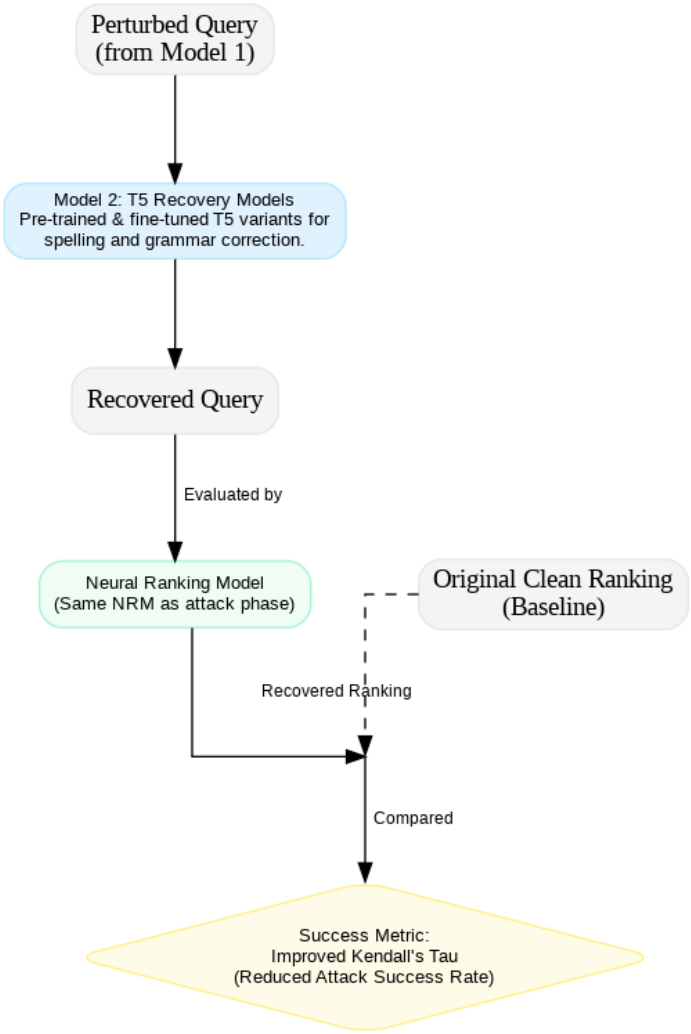

FIGURE 3.2: Framework for Model-2

via automated query restoration.

Algorithm 4: Compact Pipeline for Evaluating Adversarial Query Attack and Recovery

Input: A query set Q, a ranking model M_{rank}, an attack agent π_{attack}, a recovery model M_{recover}.
Initialize: Load all models and data. Create an empty list 'Results'.
for a set number of evaluation runs **do**
    for each original query q_{orig} in Q **do**
        Attack: Generate a perturbed query $q_{\text{pert}} \leftarrow \pi_{\text{attack}}(q_{\text{orig}})$.
        Recover: Generate a corrected query $q_{\text{recov}} \leftarrow M_{\text{recover}}(q_{\text{pert}})$.
        Measure Attack: Calculate ranking correlation τ_{pert} between rankings of q_{orig} and q_{pert}.
        Measure Recovery: Calculate ranking correlation τ_{recov} between rankings of q_{orig} and q_{recov}.
        Store key metrics $(\tau_{\text{pert}}, \tau_{\text{recov}})$ in 'Results'.
    end for
end for
Analyze: Aggregate 'Results' to compute average attack impact $(\overline{\tau_{\text{pert}}})$ and recovery effectiveness $(\overline{\tau_{\text{recov}}})$.
Output: Final statistics comparing the success of the attack vs. the defense.

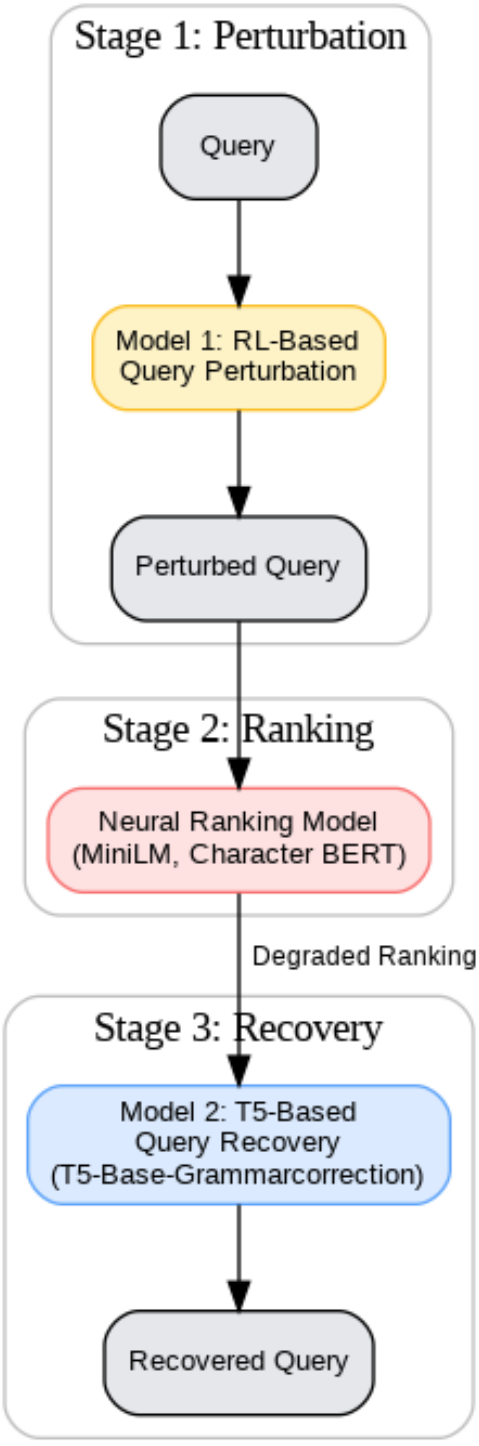

FIGURE 3.3: Proposed Framework

Chapter 4

Experiments

This chapter presents a detailed account of the experiments conducted to investigate the robustness of neural ranking models under adversarial query perturbations and the potential for recovering original queries from perturbed versions. The experimental setup is organized into three core components:

- Baseline methods
- Reinforcement Learning-based adversarial query perturbation
- Query recovery and robustness evaluation

4.1 Typo Baselines

To assess the effectiveness of our RL-based attack framework, we compare it with several baseline methods. These include both naive and heuristic approaches as well as recovery mechanisms applied post-attack.

4.1.1 Random Typo Baseline

This baseline introduces a random character-level typo in each query. The typo could be either a deletion or a swap, applied without any optimization. This method simulates accidental user input errors and serves as a naive benchmark.

- **Edit Types:** Random delete or swap (one per query).
- **Objective:** No optimization; purely random.

4.1.2 Heuristic Typo Generator

This approach uses handcrafted rules based on keyboard proximity or common spelling mistakes to generate plausible but fixed perturbations. While not optimized, these typos are more realistic than random edits.

- **Edit Types:** Substitutions based on QWERTY proximity.
- **Objective:** Rule-based typo generation.

4.2 Datasets

All experiments are conducted on the TREC Deep Learning 2019 (DL-2019) benchmark and TREC Deep Learning 2020 (DL-2020) benchmark queries, which are based on the MS MARCO Passage dataset. These benchmarks are widely used for evaluating passage retrieval models in neural IR research.

4.3 Neural Ranking Models

Model	trec-dl-2019 ndcg@10	MAP	trec-dl-2020 ndcg@10	MAP
ms-marco-MiniLM-L-6-v2	0.736	0.466	0.744	0.489
CharacterBERT	0.609	0.340	0.586	0.379
CharacterBERT+ST	0.643	0.340	0.606	0.390

TABLE 4.1: Top 1000 documents are retrieved using BM25 and reranked using the NRMs.

To simulate a real-world neural IR environment, two neural ranking models are used as black-box rankers for evaluating both perturbation impact and recovery quality. Both models were used separately to compute ranked lists for original and perturbed queries, and to measure ranking shifts (e.g., Kendall's Tau in this case) as part of the reward function.

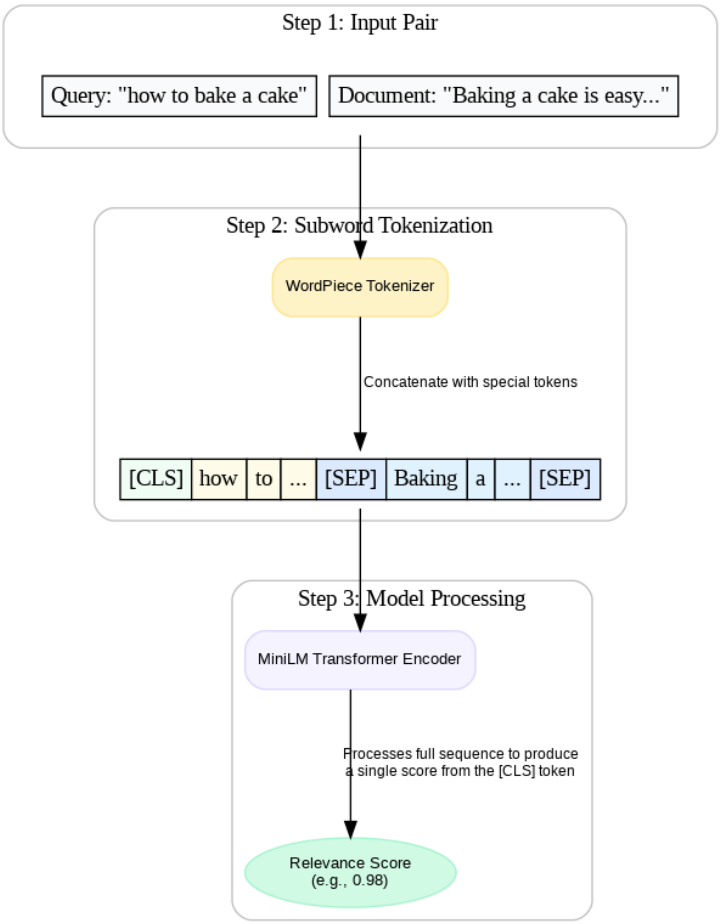

FIGURE 4.1: Overview of MiniLM

4.3.1 MiniLM Cross-Encoder

- Model: cross-encoder/ms-marco-MiniLM-L-6-v2
- Architecture: Lightweight transformer with cross-encoder architecture (query and passage jointly encoded), shown in fig 4.1
- Training: Fine-tuned on MS MARCO passage pairs for relevance ranking
- Purpose: Acts as a baseline neural ranker due to its speed and reasonable performance

4.3.2 Fine-tuned CharacterBERT

- Model: A CharacterBERT checkpoint fine-tuned on MS MARCO-style passage ranking[16]
- Architecture: Character-level transformer that processes words using character convolutional embeddings, shown in fig 4.2

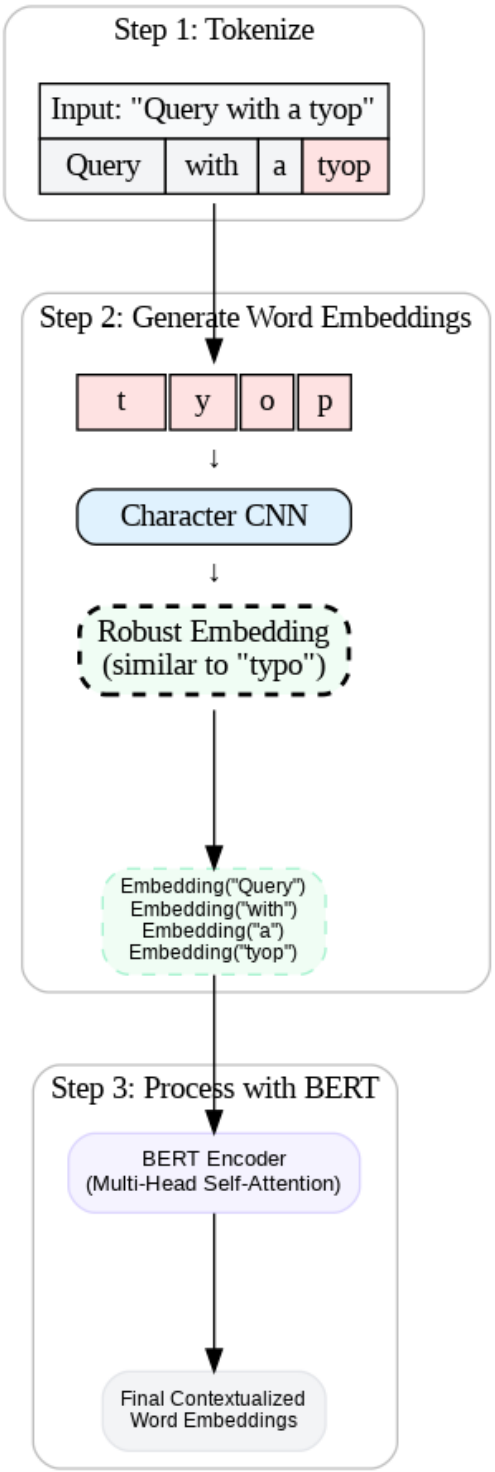

FIGURE 4.2: Overview of CharacterBERT

- Strengths: More resilient to character-level perturbations (e.g., typos), making it ideal for robustness testing
- Purpose: Used to evaluate whether character-aware models are inherently more robust to adversarial queries

CharacterBERT with Self-Teaching

To create an even more robust version of the CharacterBERT ranker, we also evaluated a variant trained with Self-Teaching (ST), a technique proposed by Zhuang and Zuccon (2022) [16]. This method is a novel adversarial training strategy designed specifically to make dense retrievers resilient to typos.

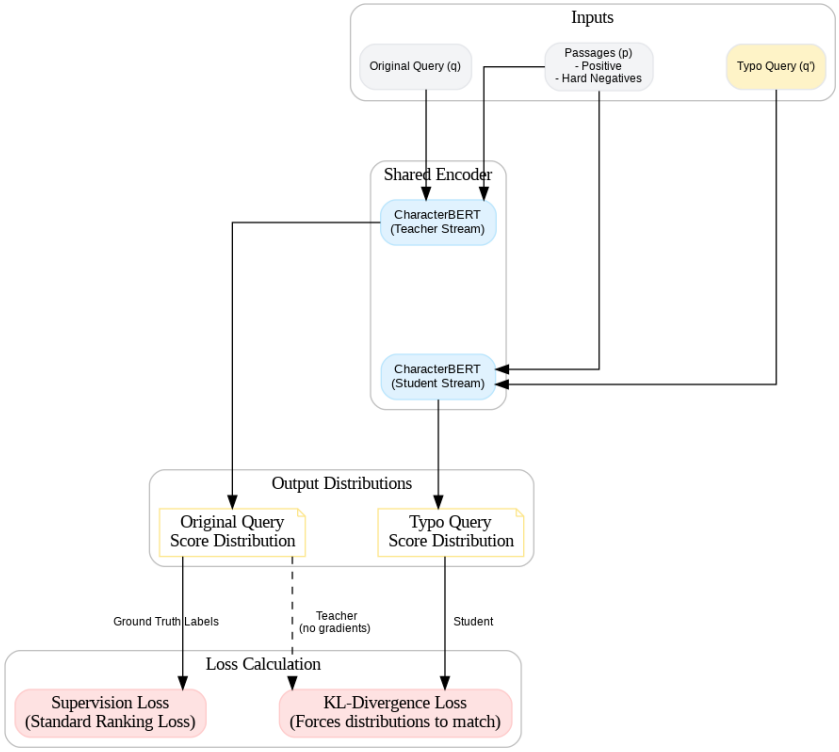

FIGURE 4.3: CharacterBERT+ST

The core idea of Self-Teaching is inspired by knowledge distillation, but with a key difference: the model acts as its own teacher. During training, for each clean query, a corresponding version with a synthetic typo is generated. The model is then trained not only on the standard ranking task but also to minimize the difference (measured by KL-divergence) between the ranking score distribution produced by the clean query and the one produced by its typo-ridden counterpart. This process, shown in fig 4.3 effectively forces the model to learn query representations that are invariant to the presence of typos, thereby enhancing its robustness beyond its inherent architectural advantages.

4.4 Reinforcement Learning Based Query Perturbation

4.4.1 RL Environment Design

- States- Each state corresponds to a tokenized form of the input query at a specific step in the perturbation process. For character-level perturbation, this is represented as a sequence of characters along with a pointer indicating the current character under consideration.
- Action Space- The action space is deliberately constrained to reflect natural human errors, ensuring realism: Swap: Swap two adjacent characters. Delete: Delete a single character. These actions are minimal and help maintain the query's semantic proximity to the original, making perturbations harder to detect while still impactful.
- Reward Function- The reward function is designed to encourage significant changes in the ranked list using Kendall's Tau distance or nDCG shift. Penalizing unnecessary or excessive perturbations, optionally incorporate linguistic knowledge, e.g.: Penalizing edits to stop words, rewarding perturbation of nouns, named entities, adjectives/adverbs. Reward function variants were tested across experiments, showing that task-aware rewards lead to more effective perturbations.

4.4.2 RL Algorithms

Three popular RL methods have been used for experiments:

- PPO (Proximal Policy Optimization): A policy-gradient method known for its stability.
- DQN (Deep Q-Network): A value-based method suited for discrete action spaces.
- A2C (Advantage Actor-Critic): A hybrid actor-critic approach offering a balance between exploration and stability.

Each agent is trained over tens of thousands of time steps (up to 50,000), with training convergence validated across three random seeds.

4.4.3 Training Setup

- Training Queries: TREC DL 2019 (43 queries from MS MARCO, via ir-datasets)
- Test Queries: TREC DL 2020 (54 queries)
- Ranking Models Used: MiniLM Cross-Encoder (cross-encoder/ms-marco-MiniLM-L-6-v2) and a fine-tuned CharacterBERT checkpoint.

The neural rankers serve as black-box oracles to evaluate ranking list changes, i.e., the RL agents do not access internal representations or gradients.

4.4.4 Results and Observations

Robustness Evaluation of MiniLM

Reward Strategy	**PPO**	**DQN**	**A2C**
Rewards only for Tau Shift	86.90	83.63	81.56
+ Penalty for Perturbing Stop Words	87.10	85.31	86.09
+ Reward for Perturbing Named Entities (NE)	87.37	86.37	89.37
+ Reward for Perturbing Nouns	87.12	**90.00**	**90.08**
+ Reward for Adjectives and Adverbs	**88.18**	89.93	88.63

TABLE 4.2: Percentage of cases in TREC DL 2020 queries for which RL agents induced a Kendall's Tau shift > 0.5 on MiniLM Cross-Encoder with various reward strategies .

The impact of each reward strategy 4.2 was evaluated on the TREC DL 2020 test queries. The results demonstrate a clear trend: agents trained with more linguistically informed and selective reward functions consistently outperformed those trained with simpler objectives. Notably, the inclusion of positive rewards for perturbing named entities and nouns yielded the highest performance. In particular, the DQN and A2C agents both surpassed 90% success rate in generating perturbations that caused a Tau shift greater than 0.5, showcasing their effectiveness in targeting semantically crucial query components.

Interestingly, PPO showed stable improvement with increasing reward complexity but peaked slightly earlier than the others, with its best performance (88.18%) under the most comprehensive reward scheme that included bonuses for adjective and adverb modifications. This suggests that while PPO benefits from richer rewards, its learning dynamics might be more sensitive to over-specification than DQN or A2C.

Robustness Evaluation of CharacterBERT

To investigate whether perturbations generated by reinforcement learning agents trained on one ranking model (MiniLM) could generalize to disrupt another model (CharacterBERT), a cross-model evaluation was conducted. A finetuned CharacterBERT model and its self-taught variant[16] were chosen due to its architectural robustness and character-level encoding, which is expected to be resilient to minor typographical errors.

Models	**PPO**	**DQN**	**A2C**
CharacterBERT	37.64	39.70	39.94
CharacterBERT+st	12.54	13.23	13.78

TABLE 4.3: Percentage of cases in TREC DL 2020 queries where the best RL agents induced a Kendall's Tau shift > 0.5 on CharacterBERT rankings.

In this setup, three best performed RL agents of PPO, DQN, and A2C—were applied to the TREC DL 2020 queries. These agents had previously been trained to generate realistic character-level perturbations (via swaps and deletions) with the goal of maximizing ranking disruption on a different neural ranker. The agents were not retrained for CharacterBERT, allowing this evaluation to measure true transferability of adversarial skill across ranking architectures.

Query Correction and Recovery

To evaluate the robustness of neural ranking systems under adversarial query perturbations, a recovery pipeline was introduced. The pipeline aimed to reverse the effects of character-level perturbations generated by reinforcement learning (RL) agents such as PPO, DQN, and A2C. These perturbations were specifically designed to disrupt ranking lists by minimally editing the original query at the character level, through operations like deletion and swap.

To mitigate this effect and attempt reconstruction of the original query, a set of pretrained language models focused on spelling and grammar correction were employed. These models were chosen due to their ability to generalize over noisy and typo-ridden text and included:

- ai-forever/T5-large-spell
- oliverguhr/spelling-correction-english-base
- vennify/t5-base-grammar-correction
- willwade/t5-small-spoken-typo

- ai-forever/sage-mt5-large

Each model was independently applied to the adversarially perturbed queries, and subsequently combined with the respective RL agents to evaluate recovery success. The outputs of these models were re-ranked using the original neural rankers, and Kendall's Tau was used to compute the similarity between the original and recovered ranked lists.

Recovery Model	**PPO (%)**	**DQN (%)**	**A2C (%)**
T5-large-spell	63.50	66.75	65.30
spelling-correction-english-base	55.15	57.75	56.90
t5-base-grammar-correction	76.25	75.15	76.66
t5-small-spoken-typo	62.50	63.90	63.75
sage-mt5-large	**52.90**	**54.75**	**54.15**

TABLE 4.4: Reduction in adversarial success rate for MiniLM Cross-Encoder after applying query recovery models

The results demonstrated that some of the individual recovery models were able to reduce the average attack success rate from over 88% (for MiniLM-based rankings) to approximately 53%. In case of CharacterBERT the average attack success rate from over 39% (for MiniLM-based rankings) to approximately 12% indicating that a significant fraction of adversarial effects were neutralized.

Recovery Model	**PPO (%)**	**DQN (%)**	**A2C (%)**
T5-large-spell	12.75	12.75	12.78
spelling-correction-english-base	14.90	15.25	15.25
t5-base-grammar-correction	37.65	36.5	38.5
t5-small-spoken-typo	12.35	12.7	13.15
sage-mt5-large	**11.65**	**12.5**	**12.7**

TABLE 4.5: Reduction in adversarial success rate for CharacterBERT after applying query recovery models

It was also observed that recovery had the most pronounced effect when the perturbed tokens involved proper nouns or critical content terms. In such cases, even partial restoration of the original token drastically reduced the ranking list disruption. However, full recovery was still challenging in queries where context or semantics were heavily altered by the perturbation.

4.5 Implementation Details

4.5.1 Dataset and Preprocessing

The TREC DL 2019 and 2020 subset of the MS MARCO passage ranking dataset has been used, accessed through the `ir_datasets` library. The following preprocessing steps have been performed:

- 43 queries have been allocated for training, and 54 for evaluation.
- For testing each query has been tested for 50 times.
- For each query, the top 25 passages from qrels have been selected and reranked.

4.5.2 Neural Ranking Model

- **MiniLM**: The `cross-encoder/ms-marco-MiniLM-L-6-v2` model has been employed as the primary neural ranker for both training the reinforcement learning agents and computing ranking metrics such as Kendall's Tau.
- **CharacterBERT**: A fine-tuned CharacterBERT checkpoint has been used during the testing phase to assess the robustness of the perturbed and recovered queries under different ranking architectures. CharacterBERT is known to be more resilient to character-level noise, making it suitable for testing query perturbation and recovery efficacy.

4.5.3 Environment Design

The environment has been implemented using the OpenAI Gym interface, with each query representing an episode. It has been configured to support multi-step interactions, combining token-level observations with character-level perturbations.

Observation Space

Each query has been represented as a 20×4 matrix comprising:

- Normalized token ID
- Stopword flag (binary)
- Named Entity (NE) flag (binary)
- Part-of-Speech (POS) flag (for NOUN, ADJ, ADV)

Padding with zero vectors has been applied to shorter queries.

Action Space

A discrete action space has been defined, comprising:

1. **Delete**: One character has been deleted at a random position.
2. **Transpose**: Two adjacent characters have been swapped.

Reward Function

The reward has been computed based on Kendall's Tau (τ) between the original and perturbed query's ranked lists:

$$\text{Reward} = (1 - \tau) \times 20 \tag{4.1}$$

This formulation has incentivized greater ranking changes with minimal perturbation.

4.5.4 Reinforcement Learning Setup

Reinforcement learning agents have been trained using implementations from the Stable Baselines3 library, specifically PPO, DQN, and A2C.

Table 4.6: Common Training Hyperparameters

Parameter	**Value**
Total Timesteps	300,000
Max Steps / Episode	5
Discount Factor (γ)	0.99
Learning Rate	3×10^{-4}
Environment Type	`DummyVecEnv`
Batch Size (PPO/A2C)	64
Replay Buffer Size (DQN)	100,000

Algorithm-Specific Settings

PPO has been configured with $n\_steps = 2048$, $clip\_range = 0.2$, $ent\_coef = 0.01$, $gae\_lambda = 0.95$.

DQN has been set up with $buffer\_size = 100{,}000$, $learning\_starts = 1000$, $exploration\_fraction = 0.1$, $exploration\_final\_eps = 0.02$, $target\_update\_interval = 500$.

A2C has used $n\_steps = 5$, $vf\_coef = 0.5$, $ent\_coef = 0.01$.

4.5.5 Training Time

Training has been conducted on a machine equipped with an NVIDIA RTX A5000 GPU (24 GB VRAM). The approximate durations required for training each agent have been summarized in Table 4.7.

TABLE 4.7: Training Time Per Agent

Agent	**Time (minutes)**
PPO	~ 450
DQN	~ 380
A2C	~ 450

4.5.6 T5-based Query Recovery Model

To reconstruct the original query from its perturbed version, T5-based sequence-to-sequence models have been used.

Evaluation of the recovery model has been conducted using:

- **Levenshtein Distance**: To quantify surface-level similarity between the recovered and original queries.
- **Kendall's Tau**: To measure whether the ranking induced by the recovered query aligns with that of the original.

Chapter 5

Conclusions and Future Work

5.1 Conclusions and Inferences

Based on the comprehensive evaluation of adversarial attacks and defenses on neural ranking models, several key conclusions can be drawn. These points collectively highlight the vulnerabilities of modern rankers, the effectiveness of the proposed attack framework, and the practical viability of a novel defense pipeline.

5.1.1 Linguistically-Informed Attacks are Significantly More Effective:

The experiments on MiniLM clearly show that a naive attack strategy is suboptimal. While an RL agent rewarded simply for any rank disruption (Kendall's Tau shift) can succeed, its performance is dramatically amplified by incorporating linguistic knowledge. By explicitly rewarding agents for perturbing high-value words like named entities and nouns, the attack success rate for A2C and DQN agents surpassed 90%. This suggests that the most potent adversarial attacks are not random but are targeted strikes on the semantic core of a query, which these agents learned to identify and exploit.

5.1.2 CharacterBERT Demonstrates Inherent Robustness, but is Not Immune:

The cross-model evaluation confirms the design principles behind character-aware models. As hypothesized, CharacterBERT is significantly more robust to character-level perturbations than the subword-based MiniLM, with the attack success rate dropping from nearly 90% to around 40% (for Top-25 lists). This resilience stems from its ability to process character sequences directly, making

it less susceptible to simple typographical manipulations. However, the fact that the success rate is still substantial indicates that inherent architectural robustness alone is not a complete defense, especially against attacks that degrade meaning. Below could ve one possible explanation for this:

vulnerable to perturbations involving spaces

The fundamental reason CharacterBERT fails on space-related perturbations is that its robustness is entirely dependent on a correct initial word tokenization step that happens before any character-level analysis takes place. The model's core assumption is that it knows what constitutes a "word". Attacks that violate this assumption break the model's workflow from the very beginning.
Token Fragmentation (Swapping with a Space) occurs when a space is introduced into a word:

- For the input "how long does it take to remove wisdom tooth", the whitespace tokenizer first splits the string into a list of three tokens: ['how', 'long', 'does', 'it', 'take', 'to', 'remove', 'wisdom', 'tooth']. The Character-CNN then processes each of these words independently to generate three robust embeddings.
- Consider the perturbed query "how long does it take to remove wisdo mtooth" (swapping 'm' and the space).
- The initial tokenizer no longer sees the token "wisdo" or "tooth". It sees ["wisdo", "mtooth"]. The Character-CNN now receives "wisdo" and "mtooth" as two separate, distinct tokens. It will generate a perfect embedding for the token "wisdo" and a perfect embedding for the token "mtooth", but it never gets to analyze the intended words "wisdom" and "tooth" as the complete words. The semantic meaning is lost before the character-level processing can even provide its benefit.

Token Fusion (Deleting a Space) occurs when a space between two words is removed.

- Normal Operation: For "definition of laudable", the tokenizer produces ['definition', 'of', 'laudable']. The Character-CNN processes 'definition', 'of' and 'laudable' separately.
- Adversarial Attack: Consider the perturbed query "definition oflaudable".
- Why it Fails: The initial tokenizer now sees only two tokens: 'definition', 'oflaudable'. It passes this new, long, and likely out-of-vocabulary word

to the Character-CNN. While the CNN can process its characters, it's analyzing a nonsensical word that it has almost certainly never encountered during training. The resulting embedding may not represent the combined meaning of 'of' and 'laudable', it will represent an arbitrary vector for a novel string of characters, causing the model's understanding to collapse.

In summary, CharacterBERT is only robust to character perturbations within the boundaries of a word. Its vulnerability is its architectural dependency on a preliminary, non-robust word tokenizer. Adversarial attacks that add or remove spaces don't target the sophisticated character-level components; they target this much simpler and more fragile pre-processing step. This is the key difference

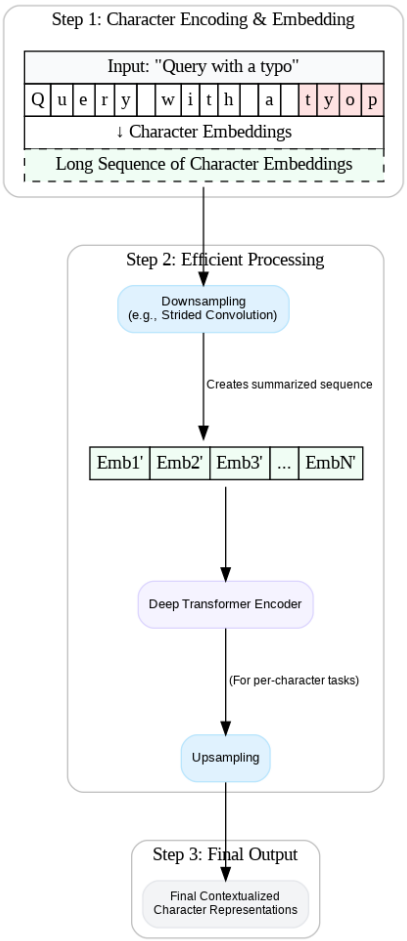

FIGURE 5.1: Overview of token-free Models

between a hybrid model like CharacterBERT and a truly token-free model 5.1 like CANINE or ByT5, which would handle these cases better because they process the entire input as a single stream of characters/bytes from the start, without a separate word tokenization step.

5.1.3 Adversarial Skills are Transferable Across Diverse Architectures:

A crucial finding is that the adversarial strategies learned on MiniLM were effective against CharacterBERT without any retraining. This demonstrates a high degree of transferability, suggesting the RL agents learned to exploit fundamental vulnerabilities in how neural models interpret textual meaning, rather than just overfitting to the weaknesses of a specific architecture. This has broader implications for the security of IR models, as it implies that an adversary could train an attack on a public model and deploy it effectively against a proprietary, black-box model with a different architecture.

5.1.4 Pre-trained Correction Models Offer an Efficient Alternative to Complex Defenses:

Many existing studies [16] propose computationally intensive methods like adversarial training, data augmentation, or self-teaching to build robust models. This work demonstrates a more efficient and practical alternative. By simply integrating off-the-shelf, pre-trained T5-based correction models into a post-processing pipeline, we achieved a significant reduction in the adversarial success rate-from over 38% down to $\sim 12\%$ for CharacterBERT.

Models	**PPO**	**DQN**	**A2C**
CharacterBERT+recovery model	11.65	12.5	12.7
CharacterBERT+self teaching	12.54	13.23	13.78

TABLE 5.1: Percentage of cases in TREC DL 2020 queries where the best RL agents induced a Kendall's Tau shift > 0.5 on CharacterBERT rankings.

This result is comparable to those achieved through explicit defensive training but is accomplished without the need for costly model fine-tuning or modification, presenting a highly practical and modular defense strategy.

5.1.5 Recovery is Most Effective on High-Impact Perturbations:

A key insight from the recovery experiments is that the defense is strongest where the attack is most potent. The T5 models were most successful at correcting perturbations involving critical content terms and proper nouns—the very same terms the RL agents found most effective to attack. This creates

an interesting dynamic: while these words are high-value targets for attackers, their semantic and orthographic uniqueness also makes them easier for a powerful language model to recognize and restore. This suggests that future defenses could be optimized by focusing resources on identifying and correcting these specific types of words.

5.1.6 A Clear Trade-off Exists Between Agent Complexity and Reward Design:

The results reveal a nuanced relationship between the choice of RL algorithm and the design of the reward function. While the A2C and DQN agents achieved peak performance by focusing on high-impact semantic components (nouns and named entities), the PPO agent's best performance (88.18%) was achieved with the most complex reward function that also included adjectives and adverbs. This suggests that different RL algorithms have varying capacities to leverage complex reward signals, and that a more sophisticated agent like PPO might be able to find effective, subtle attack vectors that simpler agents miss.

5.2 Future Work

While this study provides key insights into the vulnerability and recoverability of neural rankers under character-level adversarial attacks, several promising directions remain for future exploration:

1. Extension to Token-Level and Semantic Attacks: The current work focuses on character-level perturbations such as deletions and swaps. Future research can extend this to more sophisticated token-level or semantically misleading perturbations (e.g., synonym substitution or paraphrasing), which might be harder to detect and recover from using spelling-based correctors.

2. Adversarial Training for Robustness: One proactive defense strategy is adversarial training, where models are fine-tuned using perturbed queries generated during RL attacks. This could help the rankers learn to ignore or adapt to such distortions, improving their inherent robustness without external recovery modules.

3. future Rerankers: Our analysis revealed that while CharacterBERT is robust to intra-word perturbations, its reliance on a preliminary whitespace tokenizer makes it vulnerable to attacks involving the deletion or insertion of spaces. This dependency on correct word segmentation is a critical failure point.

A promising direction for future research is to mitigate this by employing truly token-free models, such as ByT5 [15] or CANINE [5], in a reranking capacity. These models process text as a raw stream of bytes or characters from the outset, completely bypassing the concept of word boundaries. They could be adapted to function as highly robust cross-encoder style rerankers. In this setup, a query and a candidate document would be concatenated into a single input sequence, separated by a special token. This combined stream would then be fed into the token-free model to produce a single, highly-contextualized relevance score. The key advantage of this approach would be its resilience to a wider class of perturbations, as it would be immune not only to character swaps within words but also to structural attacks that merge or split tokens by manipulating spaces.

4. User-Centric Evaluation and Usability: An important direction is to study the impact of real-world query noise (e.g., from voice input or mobile typing) and measure user satisfaction post-recovery. This could help bridge the gap between adversarial robustness research and practical IR system deployment.

Bibliography

[1] Bisk Y. Belinkov Y. "Synthetic and Natural Noise Both Break Neural Machine Translation." In: *Proceedings of the International Conference on Learning Representations (ICLR).* (2018).

[2] Hicham El Boukkouri et al. "CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters". In: *Proceedings of the 28th International Conference on Computational Linguistics* (2020).

[3] Moore R. C. Brill E. "An Improved Error Model for Noisy Channel Spelling Correction." In: *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics.* (2000).

[4] Jiang L. Macherey W. Cheng Y. "Towards Robust Neural Machine Translation." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)* (2018).

[5] Jonathan H. Clark et al. "CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation". In: *CoRR* (2021).

[6] Rao A. Lowd D. Dou D. Ebrahimi J. "HotFlip: White-Box Adversarial Examples for Text Classification." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)* (2018).

[7] Chen Q. Levy O. Gao T. "A strong baseline for sentence-level text editing." In: *Transactions of the Association for Computational Linguistics (ACL)* (2018).

[8] Ramakrishnan G. Garg S. "BAE: BERT-based Adversarial Examples for Text Classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020).

[9] Fan Y. Ai Q. Croft W. B. Guo J. "A Deep Look into Neural Ranking Models for Information Retrieval." In: *Information Processing Management.* (2019).

[10] Wieting J. Gimpel K. Zettlemoyer L. Iyyer M. "Adversarial Example Generation with Syntactically Controlled Paraphrasing." In: *Proceedings*

of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2018).

[11] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Semantically Equivalent Adversarial Rules for Debugging NLP models". In: *Association for Computational Linguistics* (2018).

[12] Sankar C. Vinyals O Rothe S. "A Simple and General-Purpose Method for Unsupervised Learning of Task-Oriented Sentence Representations." In: *Transactions of the Association for Computational Linguistics.* (2021).

[13] Zaremba W. Sutskever I. Bruna J. Erhan D. Goodfellow I. Fergus R. Szegedy C. "Intriguing properties of neural networks." In: *arXiv preprint arXiv:2004.01970.* (2013).

[14] Chen Wu et al. "PRADA: Practical Black-Box Adversarial Attacks against Neural Ranking Models". In: *arXiv:2204.01321* (2023).

[15] Linting Xue et al. "ByT5: Towards a token-free future with pre-trained byte-to-byte models". In: *Transactions of the Association for Computational Linguistics* (2022).

[16] Shengyao Zhuang and Guido Zuccon. "CharacterBERT and Self-Teaching for Improving the Robustness of Dense Retrievers on Queries with Typos". In: *SIGIR '22* (2022).