Attacking Natural Language Processing Systems With Adversarial Examples

Originally published at https://www.unite.ai/attacking-natural-language-processing-systems-with-adversarial-examples/

Researchers in the UK and Canada have devised a series of black box adversarial attacks against Natural Language Processing (NLP) systems that are effective against a wide range of popular language-processing frameworks, including widely deployed systems from Google, Facebook, IBM and Microsoft.

The attack can potentially be used to cripple machine learning translation systems by forcing them to either produce nonsense, or actually change the nature of the translation; to bottleneck training of NLP models; to misclassify toxic content; to poison search engine results by causing faulty indexing; to cause search engines to fail to identify malicious or negative content that is perfectly readable to a person; and even to cause Denial-of-Service (DoS) attacks on NLP frameworks.

Though the authors have disclosed the paper's proposed vulnerabilities to various unnamed parties whose products feature in the research, they consider that the NLP industry has been laggard in protecting itself against adversarial attacks. The paper states:

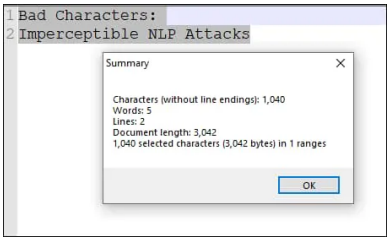
> *'These attacks exploit language coding features, such as invisible characters and homoglyphs. Although they have been seen occasionally in the past in spam and phishing scams, the designers of the many NLP systems that are now being deployed at scale appear to have ignored them completely.'*

Several of the attacks were carried out in as 'black box' an environment as can be had – via API calls to MLaaS systems, rather than locally installed FOSS versions of the NLP frameworks. Of the systems' combined efficacy, the authors write:

> *'All experiments were performed in a black-box setting in which unlimited model evaluations are permitted, but accessing the assessed model's weights or state is not permitted. This represents one of the strongest threat models for which attacks are possible in nearly all settings, including against commercial Machine-Learning-as-a-Service (MLaaS) offerings. Every model examined was vulnerable to imperceptible perturbation attacks.*
>
> *'We believe that the applicability of these attacks should in theory generalize to any text-based NLP model without adequate defenses in place.'*

The paper is titled *Bad Characters: Imperceptible NLP Attacks*, and comes from three researchers across three departments at the University of Cambridge and the University of Edinburgh, and a researcher from the University of Toronto.

The title of the paper is exemplary: it is filled with 'imperceptible' Unicode characters that form the basis of one of the four principle attack methods adopted by the researchers.



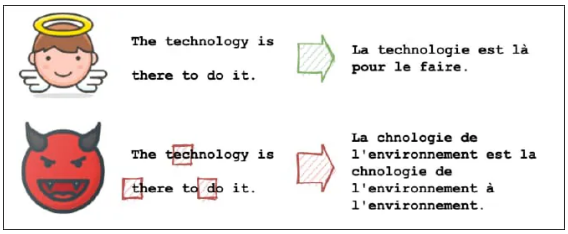
Even the paper's title has hidden mysteries.

Method/s

The paper proposes three primary effective attack methods: *invisible characters*; *homoglyphs*; and *reorderings*. These are the 'universal' methods that the researchers have found to possess wide reach against NLP frameworks in black box scenarios. An additional method, involving the use of a *delete* character, was found by the researchers to be suitable only for unusual NLP pipelines that make use of the operating system clipboard.

1: Invisible Characters

This attack uses encoded characters in a font that do not map to a Glyph in the Unicode system. The Unicode system was designed to standardize electronic text, and now covers 143,859 characters across multiple languages and symbol groups. Many of these mappings will not contain any visible character in a font (which cannot, naturally, include characters for every possible entry in Unicode).



From the paper, a hypothetical example of an attack using invisible characters, which splits up the input words into segments that either mean nothing to a Natural Language Processing system, or, if carefully crafted, can prevent an accurate translation. For the casual reader, the original text in both cases is correct. Source: https://arxiv.org/pdf/2106.09898.pdf

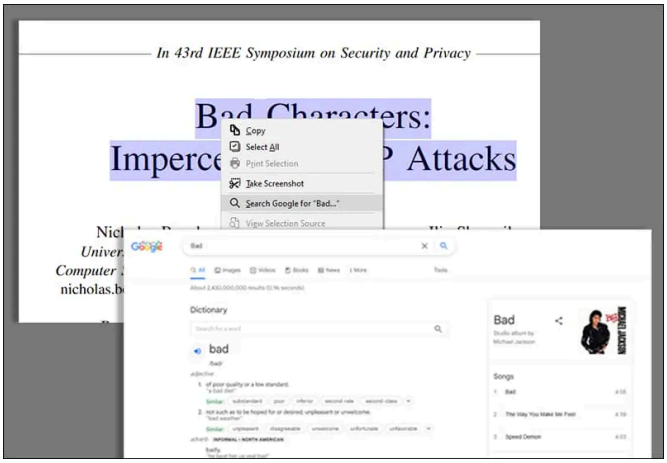
Typically, you can't just use one of these non-characters to create a zero-width space, since most systems will render a 'placeholder' symbol (such as a square or a question-mark in an angled box) to represent the unrecognized character.

However, as the paper observes, only a small handful of fonts dominate the current computing scene, and, unsurprisingly, they tend to adhere to the Unicode standard.

Therefore the researchers chose GNU's Unifont glyphs for their experiments, partly due to its 'robust coverage' of Unicode, but also because it looks like a lot of the other 'standard' fonts that are likely to be fed to NLP systems. While the invisible characters produced from Unifont do not render, they are nevertheless counted as visible characters by the NLP systems tested.

Applications

Returning to the 'crafted' title of the paper itself, we can see that performing a Google search from the selected text does not achieve the expected result:



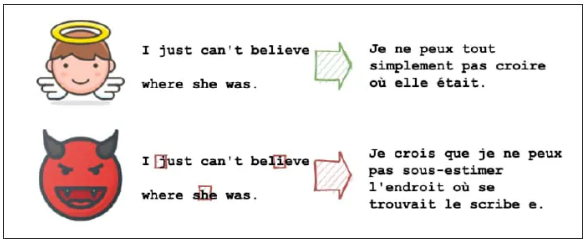
This is a client-side effect, but the server-side ramifications are a little more serious. The paper observes:

> *'Even though a perturbed document may be crawled by a search engine's crawler, the terms used to index it will be affected by the perturbations, making it less likely to appear from a search on unperturbed terms. It is thus possible to hide documents from search engines "in plain sight."*
>
> *'As an example application, a dishonest company could mask negative information in its financial filings so that the specialist search engines used by stock analysts fail to pick it up.'*

The only scenarios in which the' invisible characters' attack proved less effective were against toxic content, Named Entity Recognition (NER), and sentiment analysis models. The authors postulate that this is either because the models were trained on data that also contained invisible characters, or the model's tokenizer (which breaks raw language input down into modular components) was already configured to ignore them.

2: Homoglyphs

A homoglyph is a character that looks like another character – a semantic weakness that was exploited in 2000 to create a [scam replica] of the PayPal payment processing domain.



In this hypothetical example from the paper, a homoglyph attack changes the meaning of a translation by substituting visually indistinguishable homoglyphs (outlined in red) for common Latin characters.

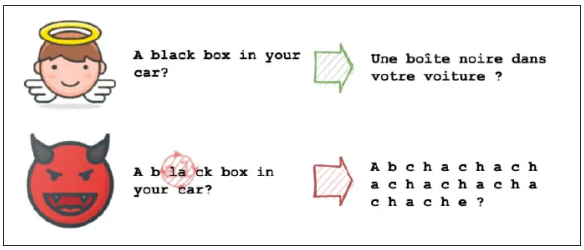
The authors comment*:

> *'We have found that machine-learning models that process user-supplied text, such as neural machine-translation systems, are particularly vulnerable to this style of attack. Consider, for example, the market-leading service [Google Translate]. At the time of writing, entering the string "paypal" in the English to Russian model correctly outputs "PayPal", but replacing the Latin character a in the input with the Cyrillic character **а** incorrectly outputs "папа" ("father" in English).'*

The researchers observe that while many NLP pipelines will replace characters that are outside their language-specific dictionary with an *<unk>* ('unknown') token, the software processes that summon the poisoned text into the pipeline may propagate unknown words for evaluation before this safety measure can kick in. The authors state that this *'opens a surprisingly large attack surface'*.

3: Reorderings

Unicode allows for languages that are written left-to-right, with the ordering handled by Unicode's Bidirectional (BIDI) algorithm. Mixing right-to-left and left-to-right characters in a single string is therefore confounding, and Unicode has made allowance for this by permitting BIDI to be overridden by special control characters. These enable almost arbitrary rendering for a fixed encoding ordering.



In another theoretical example from the paper, a translation mechanism is caused to put all the letters of the translated text in the wrong order, because it is obeying the wrong right-to-left/left-to-right encoding, due to a part of the adversarial source text (circled) commanding it to do so.

The authors state that at the time of writing the paper, the method was effective against the Unicode implementation in the Chromium web browser, the upstream source for Google's Chrome browser, Microsoft's Edge browser, and a fair number of other forks.

Also: *Deletions*

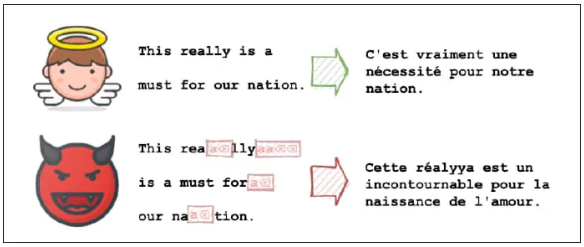
Included here so that the subsequent results graphs are clear, the *deletions* attack involves including a character that represents a backspace or other text-affecting control/command, which is effectively implemented by the language reading system in a style similar to a text macro.

The authors observe:

> *'A small number of control characters in Unicode can cause neighbouring text to be removed. The simplest examples are the backspace (BS) and delete (DEL) characters. There is also the carriage return (CR) which causes the text-rendering algorithm to return to the beginning of the line and overwrite its contents.*
>
> *'For example, encoded text which represents "Hello* **CR***Goodbye World" will be rendered as "Goodbye World".'*

As stated earlier, this attack effectively requires an improbable level of access in order to work, and would only be totally effective with text copied and pasted via a clipboard, systematically or not – an uncommon NLP ingestion pipeline.

The researchers tested it anyway, and it performs comparably to its stablemates. However, attacks using the first three methods can be implemented simply by uploading documents or web pages (in the case of an attack against search engines and/or web-scraping NLP pipelines).



In a deletions attack, the crafted characters effectively erase what precedes them, or else force single-line text into a second paragraph, in both cases without making this obvious to the casual reader.

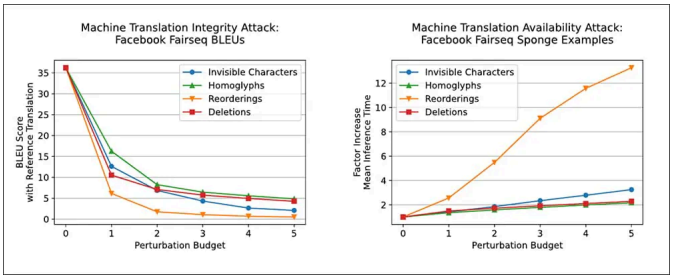
Effectiveness Against Current NLP Systems

The researchers performed a range of untargeted and targeted attacks across five popular closed-source models from Facebook, IBM, Microsoft, Google, and HuggingFace, as well as three open source models.

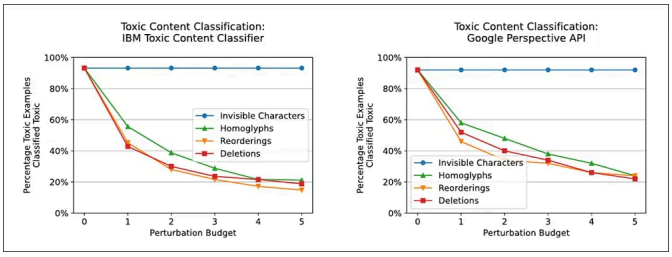
They also tested 'sponge' attacks against the models. A sponge attack is effectively a DoS attack for NLP systems, where the input text 'does not compute', and causes training to be critically slowed down – a process that should normally be made impossible by data pre-processing.

The five NLP tasks evaluated were machine translation, toxic content detection, textual entailment classification, named entity recognition and sentiment analysis.

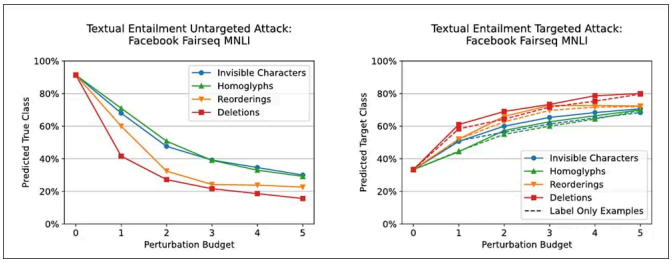
The tests were undertaken on an unspecified number of Tesla P100 GPUs, each running an Intel Xeon Silver 4110 CPU over Ubuntu. In order not to violate terms of service in the case of making API calls, the experiments were uniformly repeated with a perturbation budget of zero (unaffected source text) to five (maximum disruption). The researchers contend that the results they obtained could be exceeded if a larger number of iterations were allowed.



Results from applying adversarial examples against Facebook's Fairseq EN-FR model.

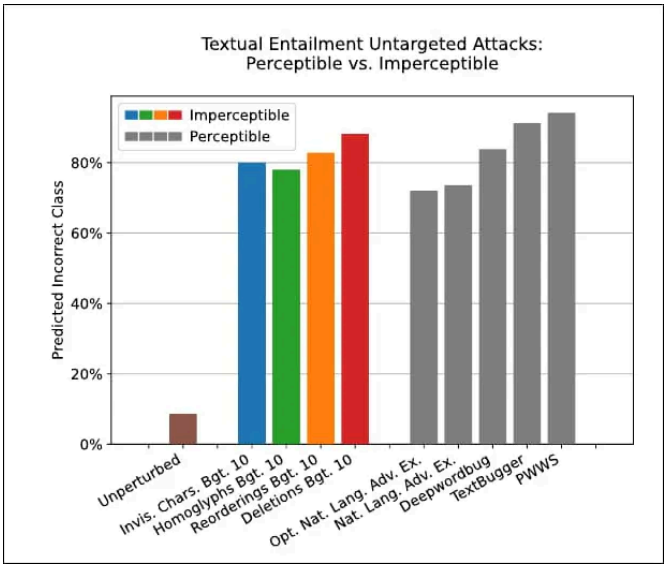


Results from attacks against IBM's toxic content classifier and Google's Perspective API.



Two attacks against Facebook's Fairseq: 'untargeted' aims to disrupt, whilst 'targeted' aims to change the meaning of translated language.

The researchers further tested their system against prior frameworks that were not able to generate 'human readable' perturbing text in the same way, and found the system largely on par with these, and often notably better, whilst retaining the huge advantage of stealth.



The average effectiveness across all methods, attack vectors and targets hovers at around 80%, with very few iterations run.

Commenting on the results, the researchers say:

> *'Perhaps the most disturbing aspect of our imperceptible perturbation attacks is their broad applicability: all text-based NLP systems we tested are susceptible. Indeed, any machine learning model which ingests user-supplied text as input is theoretically vulnerable to this attack.*
>
> *'The adversarial implications may vary from one application to another and from one model to another, but all text-based models are based on encoded text, and all text is subject to adversarial encoding unless the coding is suitably constrained.'*

Universal Optical Character Recognition?

These attacks depend on what are effectively 'vulnerabilities' in Unicode, and would be obviated in an NLP pipeline that rasterized all incoming text and used Optical Character Recognition as a sanitization measure. In that case, the same non-malign semantic meaning visible to people reading these perturbed attacks would be passed on to the NLP system.

However, when the researchers implemented an OCR pipeline to test this theory, they found that the BLEU (Bilingual Evaluation Understudy) scores dropped baseline accuracy by 6.2%, and suggest that improved OCR technologies would probably be necessary to remedy this.

They further suggest that BIDI control characters should be stripped from input by default, unusual homoglyphs be mapped and indexed (which they characterize as 'a daunting task'), and tokenizers and other ingestion mechanisms be armed against invisible characters.

In closing, the research group urges the NLP sector to become more alert to the possibilities for adversarial attack, currently a field of great interest in computer vision research.

> *'[We] recommend that all firms building and deploying text-based NLP systems implement such defenses if they want their applications to be robust against malicious actors.'*

** My conversion of inline citations to hyperlinks*

18:08 14th Dec 2021 – removed duplicate mention of IBM, moved auto-internal link from quote – MA

Writer on machine learning, domain specialist in human image synthesis. Former head of research content at Metaphysic.ai, until its dissolution into DNEG's Brahma.ai.
Portfolio site: martinanderson.ai
Contact: martin@martinanderson.ai

Discover More