Survey on Text Summarization and its Adversarial Attacks

Simone Sestito
Sapienza Università di Roma
sestito.1937764@studenti.uniroma1.it

Lorenzo Antonelli
Sapienza Università di Roma
antonelli.1888938@studenti.uniroma1.it

Abstract

In this survey, we want to explore the evolution from the past to the state-of-the-art available models for Automatic Text Summarization, observing datasets, benchmarks, and available evaluation metrics. We aim to study how models trained on news-based datasets actually perform on totally different datasets, but also if using a commercial LLM may be a better or worse idea, discovering the limits or the opportunities of these new generic models.

Finally, we want to perform some robustness checks of some recent models, based on the projected gradient method combined with group lasso and gradient regularization [4].

1 Task description

The task of compressing a piece of text into a shorter version, minimizing the size of the original text while keeping crucial informational aspects and content meaning, is known as *summarization*. [31] Most research papers in the domain of ATS (Automatic Text Summarization) have been released in the last few years, also thanks to the new Transformers architecture [30] and the increase in the available computational power. In fact, the majority of recent works are based on BERT. [27] Still, previously there were non-neural strategies such as TextRank [17] or Word Weighted Frequency [1], which could still achieve a decent result in this task.

While the topology of Text Summarization is pretty detailed, [31] we noticed that the main distinction that researchers generally make is between *Extractive* and *Abstractive* approaches. The former is responsible for summarizing the long text using some of its original sentences, while the latter aims to produce new sentences not taken from the input text itself but simulating a more natural human behavior.

1.1 Examples

To better explain what Automatic Text Summarization aims to be capable of, here we report a few samples got by some of the most famous available datasets.

1.1.1 From CNN/Daily Mail

The CNN / Daily Mail Dataset is a corpora of more than 300,000 news stories with associated summaries from the CNN and Daily Mail websites. [20].

An extract of the article text. *LONDON, England (Reuters) – Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won't cast a spell on him. [...] Meanwhile, he is braced for even closer media scrutiny now that he's legally an adult: "I just think I'm going to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast, rewritten, or redistributed.*

Output summary. *Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday . Young actor says he has no plans to fritter his cash away . Radcliffe's earnings from first five Potter films have been held in trust fund .*

1.1.2 From WikiHow dataset

This dataset contains articles written by ordinary people, not journalists, describing the steps of doing a task throughout the text. [11]

An extract of the article text. *If you're a photographer, keep all the necessary lens, cords, and batteries in the same quadrant of your home or studio. Paints should be kept with brushes, cleaner, and [...]. Don't be sentimental here. If you haven't*

used it in the last six months there is little chance you'll use it in the next six months. Toss it

Output summary. *Keep related supplies in the same area., Make an effort to clean a dedicated workspace after every session., Place loose supplies in large, clearly visible containers., Use clotheslines and clips to hang sketches, photos, and reference material., Use every inch of the room for storage, especially vertical space., Use chalkboard paint to make space for drafting ideas right on the walls., Purchase a label maker to make your organization strategy semi-permanent., Make a habit of throwing out old, excess, or useless stuff each month*

1.2 Real-world applications

The main concept behind AI text summarization is to have a short way to grasp the information contained in a long text (such as an article, a tutorial or a paper itself), since most people only need a summary and not every detail contained in the text. [5]

Daily life applications include:

- Summaries of articles in a news feed.
- Lectures notes summarization for a quick recap.
- Summarization of documents in general, when a human is scanning a large number of documents in a short time.

These applications are sometimes integrated into other automated systems. For instance, search engines started to use Extractive text summarization to directly highlight the specific citation in the original webpage that should contain the answer to our query. Moreover, chatbots started to utilize *retrieval augmented generation* to provide short descriptions of the content of a document as citations to the source document.

It is generally agreed that synthesis also entails certain disadvantages: inevitably, consuming a summary instead of the original text results in the loss of some details originally present. The most important information in some domain-specific articles may be subjective and dependent on the user's prior knowledge or specific needs. Extractive text summarization, which cites original sentences, can lose some connections between parts of the text.

2 Related work

2.1 Traditional machine learning approaches

Before the advent of the latest approaches based on Deep Learning, there were already some models based on traditional machine learning able to perform extractive text summarization.

The TextRank model (Mihalcea et al., 2004), heavily inspired by the famous PageRank [3] graph-based ranking algorithm, moved to the domain of text. When this model is applied to sentence extraction for summarization, the goal is to rank the importance of the sentences, represented as vertices in a graph. The weighted edges are representative of the similarity between sentences, measured as a function of their content overlap, in terms of the number of equal tokens, normalized in order not to promote long sentences (1). After running the usual ranking algorithm on the graph, the top-K ranked sentences are included in the summary. [17] More detailed experiments followed in order to optimize the similarity function, from equation 1 to more complex formulations such as using the cosine similarity between text vectors obtained from the representation of the documents using the classical TF-IDF. [2]

$$Sim(S_i, S_j) = \frac{|S_i \cap S_j|}{log(|S_i|) + log(|S_j|)} \quad (1)$$

Another idea that precedes the extensive use of Deep Learning is the use of the Word Frequency algorithm, applied to automatic text summarization. [1] A score is assigned to each sentence, computed as the sum of the Weighted Frequency of the words in it. At the end, the summary is obtained by the union of the sentences with a score above the threshold, which is usually the average score assigned to all sentences in the text. It can be modified, resulting in a longer or shorter automatic summary.

Both these models work on extractive text summarization, and not on abstractive one. This is because the extractive techniques are generally easier since they have to capture the importance of sentences in a corpus and then use them to generate the summary. On the contrary, abstractive summarization is more difficult because it involves a reinterpretation of the key concepts expressed in a long text.

2.2 Recurrent neural network based models

See et al. (2017) [26] introduced a hybrid Pointer-Generator network and the idea of a coverage vector to track the parts of the source text already summarized. This model combines all the strengths of both extractive and abstractive text summarization to overcome two of the main limitations of the abstractive summarization models: lack of factual accuracy and the generation of repetitive text. Both the encoder and the decoder are based on the LSTM architecture with the attention mechanism. During the decoding step, the model can decide, based on a learned probability, to generate a word from the vocabulary or to copy a word from the source text, to deal with out-of-vocabulary words.

The coverage vector, adapted from Tu et al. (2016) [28], helps to avoid the generation of repetitive text, by keeping track of the attention scores: the coverage vector at each timestep is the sum of the attention distributions over all previous decoder timesteps. The coverage vector is then passed as an additional input to the attention mechanism during the next timesteps.

2.3 BERT-based models

The BERT model architecture [6] has recently been utilized to achieve state-of-the-art performance in numerous natural language processing tasks. Its ability to create deep bidirectional representations from an unsupervised corpus has led to its use as a context encoder in some of the best-performing abstractive text summarization models.

In Zhang et al. (2019) [32] BERT is used both for the encoding and the decoding processes, thereby enhancing the quality of the summarization.

On the other hand, Miller (2019) [18], proposed an automatic Extractive text summarization technique based on extracting text embeddings from a text, more specifically lecture content, using BERT and then applying K-Means clustering to identify the closest sentences to the centroids and select them as the most meaningful. To get a sense of the performance, the pre-trained general BERT model is used, and the author supposed that having a fine-tuned version of BERT on lectures may further increase the quality of the result.

Lewis et al. (2019) [13] proposed the BART model, a denoising autoencoder for pretraining sequence-to-sequence models. It is seen as a generalization of BERT [6] and GPT [23] by the authors since it combines the power of Bidirectional encoder with the autoregressive decoder architecture. The training strategy used is to corrupt documents and being able to reconstruct them minimizing the cross-entropy reconstruction loss. The main difference is that BART is able to recover from any kind of corruption, even corrupting the entire document, making BART in this case act as a language model. Not only they evaluated BART on text summarization using both XSum (abstractive summarization) and CNN/Daily Mail (extractive summarization), but also many other NLP tasks such as Sequence Generation.

2.4 Transformers-based architectures

The Transformer architecture, introduced by Vaswani et al. (2017) [30] has become the foundation for the best-performing models for natural language processing, including BERT. Compared to RNN, Transformer-based model architecture offers significant improvements in terms of performance and efficiency, due to the self-attention mechanism.

Liu et al. (2019) [16] proposed a new encoder-decoder framework for abstractive text summarization, based on the pre-trained BertSum [15] as the Encoder and a 6-layers Transformer as the Decoder. In addition, the decoder is first fine-tuned to produce extractive summaries, while it is subsequently fine-tuned for the task of abstractive text summarization. This two-stage approach can increase the performance of the latter task.

Dong et al. (2019) [7] proposed UniLM, a modification of the Transformer architecture that unifies various language modeling tasks, such as unidirectional, bidirectional, and sequence-to-sequence, with a single Transformer framework. This approach improves the effectiveness of natural language understanding and natural language generation tasks.

Raffel et al. (2019) [24] introduced T5 (Text-to-Text Transfer Transformer), which treats all NLP tasks as a text-to-text problem, demonstrating state-of-the-art performance across a wide range of benchmarks. The authors introduced a framework that converts all text-based language problems into a text-to-text format to demonstrate the efficacy of transfer learning across many NLP tasks.

2.5 Current state-of-the-art: PEGASUS model

The PEGASUS model can achieve state-of-the-art performance in text summarization and is on par with human performance on certain datasets. The PEGASUS architecture is based on a standard Transformer encoder-decoder architecture with 568M parameters in the largest configuration: $\text{PEGASUS}_{\text{LARGE}}$.

Zhang et al. (2020) [33] discovered that masking entire sentences and then reconstructing them as the pre-training objective can significantly improve the model's performance for abstractive text summarization, requiring only a few samples for fine-tuning. The masked sentences in the text are replaced with a [MASK1] token and provided to the model with the remaining sentences. This novel pre-training objective, called Gap Sentences Generation (GSG), is designed to resemble the downstream task that the model will be required to perform and is therefore more effective than other objectives.

Selecting the proper sentences to mask from a document $D = \{x_i\}_n$ of n sentences is a challenging task that can significantly impact the final performance of the model. The authors tested three strategies: *random*, *lead*, and *principal*. The experimental results demonstrated that the principal strategy was more effective than the other strategies.

In the *principal* strategy, the ROUGE1-F1 metric between the sentence and the remaining document is used as a proxy for the importance of the sentence:

$$s_i = rouge(x_i, D \setminus \{x_i\}), \forall i \tag{2}$$

Each sentence is scored independently and the top m are selected. The authors also tested two different approaches to computing ROUGE1-F1. One approach considered the n-grams as a set, while the other approach double-counted identical n-grams. The ablation study of $\text{PEGASUS}_{\text{BASE}}$ demonstrated that the double-counting strategy performs best. As a result, this strategy was selected for $\text{PEGASUS}_{\text{LARGE}}$, the state-of-the-art performing variant.

The PEGASUS pre-training was also tested with a second objective in conjunction with GSG: Masked Language Modeling (MLM). This objective is used for BERT training and works by masking 15% of the tokens in unselected sentences. The authors observed that MLM improves fine-tuning performance at early pre-training checkpoints, but inhibits further gains with more pre-training steps. Consequently, they decided not to include this objective in $\text{PEGASUS}_{\text{LARGE}}$.

3 Datasets and benchmarks

3.1 XSum

The XSum Dataset was proposed by Narayan et al. (2018) [21] and consists of 226,711 articles published by the *BBC* between 2010 and 2017. The articles are associated with a single-sentence summary written by the article's author. The dataset covers a variety of domains.

3.2 CNN/DailyMail

The CNN/DailyMail dataset, initially proposed by Hermann et al. (2015) [10] and subsequently repurposed by Nallapati et al. (2016) [20], was designed for question-answering tasks and is one of the most popular datasets for text summarization. The dataset comprises 311,672 news articles from *CNN* and *Daily Mail*, each of which is associated with a multi-sentence summary in the form of bullet points.

3.3 NYT Annotated Corpus

The New York Times Annotated Corpus is a dataset containing more than 1.8 million articles extracted by the New York Times in 20 years, from 1987 to 2007, including article metadata. It was presented Sandhaus et al. in 2008 but, unfortunately, the number of papers using this dataset drastically dropped in the last year because it was taken down by the LDC [1] where it was previously available for download.

3.4 WikiHow

Koupaee et al. (2018) [11] proposed the WikiHow dataset, which represents one of the first large-scale datasets that do not include news articles. The dataset contains over 230,000 articles with content, title and summaries from *wikiHow*, a crowd-sourced tutorial repository written by a multitude of authors with varying writing styles, whose content can be shared the terms of CC-BY-NC-SA Creative Commons license. The summaries are the concatenation of all the bold lines (the summary sentences) of all the paragraphs.

[1] https://doi.org/10.35111/77ba-9x74

3.5 How2

The How2 dataset was introduced by Sanabria et al. (2018) [25] and consists of 2,000 hours of subtitled videos, each paired with a summary of the entire content of each of the 80,000 clips. It can be used for various NLP tasks, such as machine translation, abstractive text summarization, and automatic speech recognition.

3.6 English Gigaword

The English Gigaword dataset was introduced by Graff et al. (2003) [9] and includes more than 4 million news articles. The headline can be used as the target summary to train and evaluate text summarization models.

3.7 Benchmarks and evaluation metrics

Fabbri et al. (2021) [8] identified a lack of consensus between evaluation protocols and metrics for text summarization, which has resulted in the absence of comparable results between works.

The authors tested 23 summarization systems with the metrics and a large set of human judgments on the quality of the summaries. These judgments focused on *coherence*, *consistency*, *fluency* and *relevance*.

The authors presented a comprehensive evaluation framework that includes 14 evaluation metrics, including:

ROUGE. [14] evaluates summary quality by comparing it to a reference summary based on the overlap of n-grams. Based on the version it can use unigrams, bigrams, or the longest common subsequence of n-grams. ROUGE has different variants based on the metric considered: ROUGE-P considers the precision, ROUGE-R the recall, and ROUGE-F the F1 score.

BERTScore. [34] is a metric that evaluates summaries by comparing the similarity of word embeddings computed using BERT. This is achieved by computing the cosine similarity between the embeddings of the tokens in the candidate and reference summaries, and then aligning and scoring the tokens based on the computed similarities.

BLEU. [22] is the primary evaluation metric used in machine translation. However, it can be applied to summarization. It considers the n-gram overlap like ROUGE but incorporates a brevity penalty, to get longer and more detailed summaries.

BLANC. [29] introduces a novel approach to automatically assess summary quality by measuring how much BERT improves on a document comprehension task when given access to the summary. Unlike other scoring methods, the BLANC metric does not require the use of human-written reference summaries. The usefulness of the document is used as a proxy for the summary quality.

MoverScore. [35] combines BERT's contextual representation of the summary and reference text, along with their semantic distance, utilizing Word Mover's Distance [12]. The n-gram representations are weighted through IDF and then pooled with the Power Means. The resulting score exhibits a strong correlation with human evaluation.

4 Existing tools, libraries, papers with code

It is good to know that most of the previously discussed papers and works in section 2 have been implemented and their source code publicly released, even if not directly by the original authors of a specific paper sometimes.

The vast majority of the projects discussed here based on Deep Learning techniques have been implemented with PyTorch or TensorFlow. Moreover, they are available on HuggingFace and can be integrated in other experiments using the `transformers` library.[2]

Specifically, the lecture summarizer project [18] already comes with a Docker container for reproducibility and a live demo making possible its usage even without running code locally.[3].

We decided not to run experiments on the UniLM [7] model since it may require too many resources to properly be run, and also the project proposed in Zhang et al. (2019) [32] because we have only been able to find an unofficial implementation, [4] also missing some parts and explicitly stating that it requires many GBs of GPU memory to run the full model, making experiments unfeasible.

Evaluation tools. Speaking about the evaluation libraries, we plan to use the project SummEval [8], which allows to run a variety of evaluations at the same time, using multiple metrics. In order to make this project run properly, we had to containerize it by ourselves, including all the necessary dependencies and Python version. Since this

[2] https://huggingface.co/docs/transformers/index

[3] https://smrzr.io/

[4] https://github.com/nayeon7lee/bert-summarization

process was not straightforward, we plan to send a contribution to the original SummEval repository [5] to make it more usable for everyone in the future.

Standard ML approaches. When it comes to the algorithmic approaches and the standard machine learning models to perform extractive text summarization ([1], [17]), they are easily implementable using `numpy`[6] to seamlessly work with tensors and standard NLP libraries, such as `NLTK`[7], to use their implementations of a sentence tokenizer, stop words removal, and other basic fundamental operations. For the TextRank comparison, we will adopt the implementation with Levenshtein Distance as the relation between text units, made available on GitHub.[8]

Datasets. As mentioned in section 3.3, the NYT dataset is unavailable so it will not be included at all in our experimental setting. Most of other datasets are made available by the HuggingFace `datasets` library [9]. This makes the retrieval and usage much easier, letting us focus on the results. We observed that some datasets such as WikiHow [11] are not automatically retrieved by the HuggingFace library, but have to be manually downloaded. Finally, the How2 dataset [25] requires to fill a form to submit to the authors to be able to download and use it.

Adversarial Attacks. To explore the field of Adversarial Attacks applied to the NLP task of automatic text summarization, our goal is to make the model hallucinate and provide a different summary not reflecting the actual content of the input text, by finding a perturbation of the corpus such to induce this bad behaviour. On the contrary of other artificial intelligence fields, having a continuous input space such as images or audio, we not only have to define what we mean as an *imperceptible perturbation of the input*, but also how to apply it to a discrete input space where text lives in.

Some papers in this direction also provided code implementations to make this test easier to be run, in the difficult setting of text input. Cheng et al. in 2018, with the `Seq2Sick` [4] paper, presented a first idea on how to address the challenges that come with working in a discrete input space with sequence-to-sequence models. They implement a projected gradient method combined with group lasso and gradient regularization. In the available code on GitHub[10] they provide a Python tool to perform automatic attacks on many models, given in input as `.pt` files, which are the weights and biases of a model in PyTorch.

`TextAttack` [19][11] (Morris et al., 2020) propose an extensible Python framework for adversarial attacks, data augmentation, and model training in NLP. By *extensible* we mean that it creates a framework to write our own *recipes*, which implement specifically crafted attacks, including known attacks from the literature already provided by the authors or the community.

5 State-of-the-art evaluation

In the original work [33] $\text{PEGASUS}_{\text{LARGE}}$ was evaluated on 12 different datasets, including *CNN/DailyMail*, *Gigaword*, *WikiHow* and *XSum*. These datasets are discussed in section 3. The authors measured the ROUGE1, ROUGE2, and ROUGEL scores and compared the results with different models with three datasets. The models included *UniLM* and *BART*, discussed in section 2. $\text{PEGASUS}_{\text{LARGE}}$ showed state-of-the-art performance compared to other models across all the three datasets used for the comparison.

In addition, $\text{PEGASUS}_{\text{LARGE}}$ was evaluated in a low-resource scenario, to assess its performance in a context similar to real-world use cases. The model outperformed previous state-of-the-art results on six out of twelve datasets with only 1000 fine-tuning samples.

On the CNN/DailyMail dataset, $\text{PEGASUS}_{\text{LARGE}}$ performed better than GPT-2, a model with double the number of parameters. In both zero-shot and with 1000 samples, $\text{PEGASUS}_{\text{LARGE}}$ demonstrated better performance in terms of ROUGE2-F.

6 Comparative evaluation

6.1 Datasets

To run this comparative evaluation we decided to pick 3 different datasets:

1. **WikiHow**, the dataset composed by articles from the famous website, because it is dif-

[5] https://github.com/Yale-LILY/SummEval/pull/54
[6] https://numpy.org/
[7] https://www.nltk.org/
[8] https://github.com/davidadamojr/TextRank
[9] https://huggingface.co/docs/datasets/index
[10] https://github.com/cmhcbb/Seq2Sick
[11] https://github.com/QData/TextAttack

ferent from usual news datasets and it is not often used

2. **XSum**, the most popular news summarization dataset, since a lot of models are trained on it

3. **CNN/DailyMail**, an alternative news dataset to do a comparison between the two

6.2 Models evaluated

We have run the following models (in alphabetic order):

1. **BART**: sequence-to-sequence model with an encoder and a decoder

2. **BERT K-Means**: extractive algorithm based on K-Means executed on top of BERT sentence embeddings

3. **Claude 3 Haiku**: we also wanted to compare the performances of a cheap LLM with specialized models in order to observe how well an LLM behaves, without any fine-tuning, but just a task description in natural language sent as the system context

4. **ExtractiveBasic**: simplest algorithm based on finding the most relevant sentences in a text corpus, computing their score using TF-IDF

5. **Pegasus**: sequence-to-sequence model with the same encoder-decoder model architecture as BART, made by Google, trained on XSum

6. **Pegasus Large**: the Large variant of the Pegasus model, trained on XSum

7. **TextRank**: adaption of the PageRank algorithm for text summarization - this implementation [12] uses the Levenshtein edit distance, instead of the standard formula discussed earlier (1)

Deep Learning models have been downloaded from HuggingFace, while other models have an implementation available on GitHub, or they are easily implementable.

[12] https://github.com/davidadamojr/TextRank

6.3 Metrics and evaluation protocol

Since we are including in the evaluation also some heavy models such as Pegasus Large or BERT-based models, and given the limited computational resources available, we have chosen to limit the samples to feed to the models to 2000 per dataset, extracted from the test split, using the same samples for every model to have a fair comparison.

Specifically for the use of the LLM, we used the following description passed in the system context, [13] not directly in the user prompt:
You are a text summarization machine. The following text is a long text, and I want you to output only the text of a short summary of it. Use at most 200 characters for your output. Do not refer to the text in third person. Start directly with the summary content.

The execution of the LLM, being provided as a REST service, it comes with its own usage costs. Specifically, we can see the input and output tokens (fig. 1), as well as the final cost (fig. 2). It is also useful to know to evaluate whether the adoption of a generic LLM may have other advantages from this point of view too, when compared to the integration of other ad-hoc models.

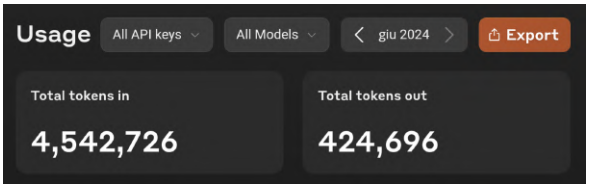

Figure 1: Total tokens sent and received from Claude 3 Haiku

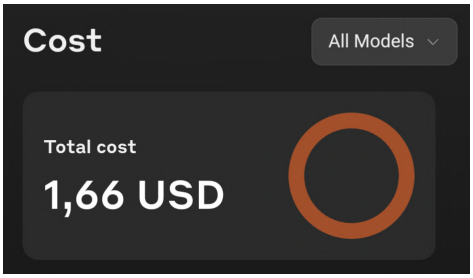

Figure 2: Total cost of the Claude 3 Haiku execution for this experiment

After collecting the results for each sample of the dataset, and running every model, we computed the following metrics:

[13] https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/system-prompts#how-to-give-claude-a-role

1. **ROUGE**, the most popular text summarization metric, in variants `-1`, `-2` and `-L`, reporting Precision, Recall and F1-score; computed using the SummEval toolkit

2. **BERTScore**, a metric that takes into account the similarities between token and sentence embeddings, using the model output and the reference summary

3. **MoverScore**, a benchmark with Contextualized Embeddings and Earth Mover Distance, used for many NLP text generation tasks

We excluded some metrics from our evaluation, for different motivations:

- **BLEU**, since the implementation we found takes as input paired sentences, and this is a metric specifically created for text translation, we could not apply it for text summarization, where having paired sentences is not possible [14]

- **BLANC**, even if this metric looked the most interesting, it was excluded for computational reasons: in the official GitHub page of the project, the authors stated that running on a NVIDIA V100 GPU it may take around 1.4 sec per summary. From a quick calculation, we can see that with 6000 samples $\times$ 7 models makes it unfeasible to be run

6.4 Results and discussion

While observing the results, we have to take into account that, while specialized models have never observed the samples we used to run the evaluations, the LLM, being a general model trained on the Internet, may have those samples in its own training set.

In appendix A detailed results are reported.

We can observe that on CNN/DailyMail (1) and WikiHow (2) datasets, the LLM always scores the best F1. This is counterbalanced by the fact that it is a closed-source model, with many more parameters, having a much higher execution cost. Finally, we cannot know the impact that the fact these datasets are publicly available and famous, presumably being included in the training sets of these LLMs, may have had in the evaluation.

Still, an open discussion may be if specialized NLP models can still have a leading role for specific tasks without fine-tuning. In fact, we can observe a different situation in the case of the XSum dataset. In this case the Pegasus model has the best results, with BART in second place and Claude 3 Haiku only at third place. However, a future work may be to compare the performances against more powerful available LLMs: we have chosen to use the absolute cheapest one available on the market.

When it comes to the length of the output summaries, Pegasus and BART are always the top models in this sense, halving the long input text (4), and being the only ones producing a summary shorter than the reference one (5).

7 Adversarial Attacks

7.1 TextAttack

TextAttack [19] provides a convenient Python framework for performing adversarial attacks on sequence classification and sequence-to-sequence models and provides an implementation of Seq2Sick among the attack recipes[15].

One of the problems we immediately noticed is that the framework does not provide, out-of-the-box, support for sequence-to-sequence HuggingFace models, since in the code it uses *AutoModelForSequenceClassification* for all the HuggingFace's models. This issue does not occur using the officially supported models in the *Model's Zoo*. We have identified and corrected this problem and added support for all sequence-to-sequence models in the HuggingFace transformer library.

The second problem we found is that TextAttack uses a greedy re-implementation of Seq2Sick, which does not use gradient descent. In our experiments, this led to suboptimal results and an extremely low success rate with all the models we tested.

We therefore decided to switch to a different implementation of Seq2Sick that is closer to the original, which allowed us to obtain better results.

7.2 Seq2Sick Attack

Cheng et al. (2018) [4] proposed a novel framework for crafting adversarial examples on sequence-to-sequence models. Their approach

[14] https://github.com/mjpost/sacrebleu/blob/master/sacrebleu/compat.py#L9

[15] https://textattack.readthedocs.io/en/master/_modules/textattack/attack_recipes/seq2sick_cheng_2018_blackbox.html#Seq2SickCheng2018BlackBox

combines a *projected gradient* method combined with *group lasso* and *gradient regularization*. To address the discrete input space, the authors introduced a *non-overlapping loss function*. At each iteration, the current adversarial output $x_i + \delta_i$ is projected back into $\mathbb{W}$, the set of word embeddings of all words in the input vocabulary, to ensure that $X + \delta$ can map to a specific input word. The non-overlapping loss function is defined as:

$$L_{\text{non-overlapping}} = \sum_{t=1}^{M} \max\{-\epsilon, z_t^{(s_t)} - \max_{y \neq s_t}\{z_t^{(y)}\}\}$$

Where ϵ is the confidence margin, s_t is the t-th word in the output vocabulary, and z_t is the logit layer output of the adversarial example.

Typically, in adversarial attacks, the ℓ_2 norm is used to measure the distortion. However, for this task, it is not suitable, since the vast majority of the learned $\{\delta_t\}_{t=1}^{M}$ using the ℓ_2 norm will be non-zero. Consequently, the adversarial example will be significantly different from the input sentence. To solve this issue the authors treated each δ_t as a group, applying then the group lasso regularization:

$$R(\delta) = \sum_{t=1}^{N} ||\delta_t||_2$$

Finally, to penalize large distances in $\mathbb{W}$ between the adversarial example and the other embedding vectors, the authors used a gradient regularization, defined as:

$$\sum_{i=1}^{N} \min_{w_j \in \mathbb{W}} \{||x_i + \delta - w_j||_2\}$$

7.3 Results

We applied the Seq2Sick attack to three models: *T5$_{SMALL}$*, *PEGASUS* and *PEGASUS$_{LARGE}$* with varying results.

The Se2Sick attack gave us promising results on T5$_{\text{SMALL}}$, with completely non-overlapping results, single words, or spaces. Table 6 highlights some of the adversarial examples we have generated.

Regarding PEGASUS and PEGASUS$_{\text{LARGE}}$, the results were mixed. In the case of shorter sentences, some successful adversarial examples were crafted, as highlighted in table 7. With longer sentences, both models appear to demonstrate some degree of resistance, and we failed to craft adversarial examples that generate a completely non-overlapping output, as shown in table 8.

All detailed results are available in the GitHub repository, and the most significant results are reported in appendix C.

7.4 Discussion

One of the possible reasons why we obtained a lower success rate than in the original paper is that Seq2Sick was born as an adversarial attack for RNN and LSTM models. The Transformer architecture, as shown in the original paper, is inherently more robust to these attacks [4].

In addition to that, it was observed that larger models are more robust to attacks than smaller models. T5$_{\text{SMALL}}$, with 60 million parameters, was found to be less robust than both PEGASUS, with 223 million parameters, and PEGASUS$_{\text{LARGE}}$, with 568 million parameters. Both PEGASUS models showed a different degree of robustness according to sentence length. This is probably due to the way they were trained. Both models were pre-trained on entire newspaper articles, and in our experiments, they showed worse performance with shorter sentences, even occasionally showing signs of hallucinations. This is probably the main reason that makes this attack easier with shorter sentences, as a minor perturbation is sufficient to generate a completely non-overlapping output, often meaningless.

8 Conclusions

We explored the evolution and current state-of-the-art models in the task of Automatic Text Summarization. We also have questioned about the use of a generic commercial LLM instead of a specialized model, resulting generally in better results when applied with no available datasets and no fine-tuning. However, ad-hoc models are still better when we provide a good task-specific dataset to train them on, when compared to cheap LLMs.

Moreover, the other models do not seem to have a strong bias towards the dataset they have been trained on, which is XSum in most of the cases. Indeed, their performances do not noticeably decrease when they are used, without any fine-tuning, on totally different datasets. This confirms their quality and capacity for generalisation.

Finally, we also observed that most recent models, such as Pegasus, do not appear to be attackable by the available Adversarial attacks such as Seq2Sick, making them more robust than older, smaller, models.

Future works. With respect to the Adversarial attacks, it would be interesting for the future to see how attackable models specialized in Text Summarization would react when several of those input perturbations get added to the dataset. Related work in the continuous input space shows that it is a commonly used technique to drastically improve the model robustness, and it would be captivating to perform these experiments on the discrete text input space.

Source code. The source code for our attacks (section 7) and comparative evaluations (section 6) are available on a public and shared GitHub repository: https://github.com/simonesestito/NLP-Adversarial-Text-Summarization.

Contributions.

- Simone Sestito
  - Introduction to the topic
  - Half of the models in section 2, with a bit more focus on BERT-based models and traditional machine learning
  - Half of the datasets in section 3
  - Tools, implementation details and available code in section 2, including the containerization of SummEval project to make it more usable out-of-the-box and setup the infrastructure in 6
  - Adversarial attacks in section 7
- Lorenzo Antonelli
  - Half of the models in section 2, with a bit more focus on BERT-based models and Transformers-based architectures
  - Half of the datasets in section 3
  - Evaluation in section 5
  - Half of the work on the explanation in Adversarial attacks 7
  - Part of the models in the comparative evaluation (section 6)
  - Adversarial attacks

As can be deduced from this short list, the project has been divided as equally as possible for each section, especially the *most challenging* ones.

References

[1] Anbukkarasi Abinaya and Varadhaganapathy. Extractive text summarization using word frequency algorithm for english text. 2022. 1, 2, 6

[2] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606, 2016. 2

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. Proceedings of the Seventh International World Wide Web Conference. 2

[4] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128, 2018. 1, 6, 8, 9

[5] Virender Dehru, Pradeep Tiwari, Gaurav Aggarwal, Bhavya Joshi, and Pawan Kartik. Text summarization techniques and applications. *IOP Conference Series: Materials Science and Engineering*, 1099:012042, 03 2021. 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. 3

[7] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation, 2019. 3, 5

[8] Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. Summeval: Re-evaluating summarization evaluation, 2021. 5

[9] David Graff and Christopher Cieri. English gigaword. LDC2003T05, 2003.

https://catalog.ldc.upenn.edu/LDC2003T05. 5

[10] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend, 2015. 4

[11] Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305, 2018. 1, 4, 6

[12] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org, 2015. 5

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. 3

[14] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. 5

[15] Yang Liu. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318, 2019. 3

[16] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345, 2019. 3

[17] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics. 1, 2, 6

[18] Derek Miller. Leveraging BERT for extractive text summarization on lectures. *CoRR*, abs/1906.04165, 2019. 3, 5

[19] John X. Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. Textattack: A framework for adversarial attacks in natural language processing. *CoRR*, abs/2005.05909, 2020. 6, 8

[20] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016. 1, 4

[21] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018. 4

[22] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. 5

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets researchcovers/languageunsupervised/language understanding paper. pdf*, 2018. 3

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. 3

[25] Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. How2: A large-scale dataset for multimodal language understanding, 2018. 5, 6

[26] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017. 3

[27] Dima Suleiman and Arafat Awajan. Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges. *Mathematical Problems in Engineering*, 2020, Aug 2020. 1

[28] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In Katrin Erk and Noah A. Smith, editors, *Proceedings*

of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics. 3

[29] Oleg V. Vasilyev, Vedant Dharnidharka, and John Bohannon. Fill in the BLANC: human-free quality estimation of document summaries. *CoRR*, abs/2002.09836, 2020. 5

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 1, 3

[31] Divakar Yadav, Jalpa Desai, and Arun Kumar Yadav. Automatic text summarization methods: A comprehensive review, 2022. 1

[32] Haoyu Zhang, Jianjun Xu, and Ji Wang. Pretraining-based natural language generation for text summarization, 2019. 3, 5

[33] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2020. 4, 6

[34] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. 5

[35] Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance, 2019. 5

Appendices

A Evaluation results

Model / Metric	ROUGE-1 (precision/recall/F1)	ROUGE-2 (precision/recall/F1)	ROUGE-L (precision/recall/F1)	BERTScore (precision/recall/F1)	MoverScore
BART	**0.3891**/0.2115/0.2636	0.1242/0.0649/0.0818	**0.275**/0.1498/0.1864	**0.8827**/0.8444/0.8629	0.5470
BERT K-Means Extractive	0.2003/0.5559/0.2775	0.075/0.1992/0.1023	0.1285/0.357/0.1777	0.8343/0.8735/0.8533	0.5503
Claude 3 Haiku	0.3418/0.4286/**0.3660**	0.1247/0.1563/**0.1332**	0.2260/0.2873/**0.2433**	0.8786/0.8738/**0.8760**	**0.5724**
ExtractiveBasic	0.1089/**0.7486**/0.1814	0.0496/**0.3323**/0.0823	0.0743/**0.5143**/0.1238	0.8097/**0.881**/0.8437	0.5366
Pegasus	0.377/0.2126/0.2566	**0.1346**/0.0749/0.0907	0.2708/0.1518/0.1832	0.877/0.8423/0.8591	0.5434
Pegasus Large	0.2583/0.4441/0.3017	0.0999/0.1614/0.113	0.1715/0.2912/0.1985	0.8483/0.8641/0.8559	0.5535
TextRank	0.1843/0.4826/0.2604	0.06/0.1577/0.0847	0.1107/0.2951/0.1571	0.8304/0.8618/0.8457	0.5422

Table 1: Model scores on the CNN/DailyMail test dataset

Model / Metric	ROUGE-1 (precision/recall/F1)	ROUGE-2 (precision/recall/F1)	ROUGE-L (precision/recall/F1)	BERTScore (precision/recall/F1)	MoverScore
BART	**0.3439**/0.1418/0.1799	**0.0806**/0.0307/0.0393	**0.2503**/0.1056/0.1325	**0.8697**/0.849/0.859	0.5316
BERT K-Means Extractive	0.1984/0.4622/0.2496	0.0477/0.1172/0.061	0.1239/0.3007/0.1578	0.8379/0.8741/0.8555	0.5456
Claude 3 Haiku	0.3035/0.3752/**0.3027**	0.0754/0.0994/**0.0771**	0.2035/0.2652/**0.2077**	0.8656/**0.8827**/**0.8738**	**0.5622**
ExtractiveBasic	0.12/**0.6516**/0.1839	0.0354/**0.2144**/0.0558	0.0782/**0.4594**/0.122	0.8165/0.8755/0.8447	0.5393
Pegasus	0.32/0.12818/0.16172	0.0714/0.0274/0.0346	0.2432/0.0979/0.1228	0.8638/0.8463/0.8548	0.5283
Pegasus Large	0.2281/0.348/0.2417	0.0517/0.082/0.055	0.144/0.227/0.1533	0.8423/0.8648/0.8532	0.5418
TextRank	0.1876/0.4478/0.2402	0.0412/0.108/0.054	0.1063/0.2761/0.1396	0.83/0.8686/0.8487	0.5396

Table 2: Model scores on the WikiHow test dataset

Model / Metric	ROUGE-1 (precision/recall/F1)	ROUGE-2 (precision/recall/F1)	ROUGE-L (precision/recall/F1)	BERTScore (precision/recall/F1)	MoverScore
BART	0.4853/0.4342/0.4513	0.2398/0.2129/0.2221	0.3988/0.3574/0.3713	0.9212/0.9119/0.9164	0.6101
BERT K-Means Extractive	0.1129/0.4089/0.1688	0.0174/0.0673/0.0264	0.0733/0.2668/0.1096	0.8272/0.8719/0.8489	0.5293
Claude 3 Haiku	0.1821/0.4076/0.2471	0.0456/0.1052/0.0624	0.1255/0.2826/0.1706	0.8596/0.8878/0.8733	0.5489
ExtractiveBasic	0.0725/**0.5498**/0.1198	0.0142/0.1273/0.0242	0.0486/**0.3745**/0.08	0.8093/0.8737/0.8401	0.5207
Pegasus	**0.5129**/0.4401/**0.4638**	**0.2651**/**0.2255**/**0.2388**	**0.4275**/0.3661/**0.3864**	**0.9246**/**0.9129**/**0.9186**	**0.6141**
Pegasus Large	0.1386/0.3236/0.1770	0.0223/0.0563/0.0289	0.0950/0.2174/0.1201	0.8375/0.8665/0.8516	0.5283
TextRank	0.0986/0.4453/0.1596	0.0164/0.0766/0.0267	0.0633/0.2881/0.1025	0.8205/0.8701/0.8445	0.5259

Table 3: Model scores on the XSum test dataset

B Summaries length

Model / Dataset	CNN / DailyMail	XSum	WikiHow	*Overall*
BART	**95.4% ± 2.6%**	**91.3% ± 13.2%**	**94.6% ± 9.1%**	**93.8% ± 9.6%**
BERT K-Means Extractive	76.2% ± 4.9%	71.3% ± 13.2%	75.6% ± 9.6%	74.4% ± 10.1%
Claude 3 Haiku	88.3% ± 6.4%	77.2% ± 31.7%	82.3% ± 15.9%	82.6% ± 21.3%
ExtractiveBasic	38.1% ± 8.3%	40.3% ± 9.7%	40.2% ± 8.3%	39.6% ± 8.8%
Pegasus	**94.9% ± 4.1%**	**91.3% ± 17.6%**	**94.7% ± 8.8%**	**93.6% ± 11.7%**
Pegasus Large	84.6% ± 6.3%	81.7% ± 9.0%	82.7% ± 10.0%	83.0% ± 8.7%
TextRank	75.4% ± 15.1%	58.3% ± 28.0%	66.8% ± 25.1%	66.8% ± 24.4%

Table 4: Output summary length, expressed in percentage with respect to the input text length. Higher means shorter summaries, which is better

Model / Dataset	CNN / DailyMail	XSum	WikiHow	*Overall*
BART	**44.5% ± 24.2%**	**9.6% ± 27.2%**	**50.6% ± 46.5%**	**34.9% ± 38.6%**
BERT K-Means Extractive	-249.3% ± 205.5%	-341.9% ± 340.5%	-195.5% ± 197.8%	-262.2% ± 263.5%
Claude 3 Haiku	-42.7% ± 59.0%	-151.2% ± 97.0%	-68.2% ± 125.2%	-87.4% ± 108.0%
ExtractiveBasic	-868.0% ± 660.7%	-1007.3% ± 1055.5%	-678.0% ± 564.6%	-851.1% ± 800.8%
Pegasus	**41.0% ± 39.3%**	**9.6% ± 51.1%**	**52.1% ± 50.8%**	**34.2% ± 50.7%**
Pegasus Large	-111.0% ± 118.8%	-186.7% ± 206.6%	-103.5% ± 168.9%	-133.7% ± 172.8%
TextRank	-183.1% ± 104.2%	-381.9% ± 173.1%	-220.4% ± 234.7%	-261.8% ± 198.5%

Table 5: Output summary length, expressed in percentage with respect to the reference text length. Higher means shorter summaries, which is better

C Adversarial attacks results

Source input sequence	portlet, you must write three short deployment descriptors: web.xml, portlet.xml, and geronimo-web.xml. (some of these may have been generated by your ide.)
Adversarial input sequence	portlet, you must write three short deployment descriptors: web.xml, port **suprafete**.xml, and geronimo-web.xml. (some of these may have been generated by your ide.)
Source output sequence	let, you must write three short deployment descriptors: web.xml, portlet.xml, and geronimo-web.xml. (some of these may have been generated by your ide.
Adversarial output sequence	” ”
Source input sequence	\”the chinese super league starts in a couple of weeks and sheffield united have asked me to go over and have a look at the coaching set-up, \” said mckinna.
Adversarial input sequence	\”the chinese super league starts in a couple of weeks and sheffield united have asked me to go over and have a look at the coaching set-up, \” said **mckintindea**.
Source output sequence	mckinna said sheffield united have asked me to go over and have a look at the coaching set-up. mckinna said the chinese super league starts in a couple of weeks
Adversarial output sequence	mckintindea.
Source input sequence	brain: if you don’t mind, jonathan, while you and mr. sun get acquainted, i’d like to check the arrangements for the meeting.
Adversarial input sequence	brain:**rosii** you don’t mind, jonathan, while you and mr. sun get acquainted, i’d like to check the arrangements for the meeting.
Source output sequence	: if you don’t mind, jonathan, while you and mr. sun get acquainted, i’d like to check the arrangements for the meeting.
Adversarial output sequence	jonathan, ich w\u00fcnsche mir eine erfreuliche Begegnung mit dem jungen jungen jungen jungen jungen [...]

Table 6: Adversarial examples on $T5_{SMALL}$ model

Source input sequence	portlet, you must write three short deployment descriptors: web.xml, portlet.xml, and geronimo-web.xml. (some of these may have been generated by your ide.)
Adversarial input sequence	**INRDealslet**, you must write three short deployment descriptors: web.xml, portlet.xml, and geronimo-web.xml. (some of these may have been generated by your ide.)
Source output sequence	portlet, you must write three short deployment descriptors: web, portlet by your ide, and geronimo-web.
Adversarial output sequence	The Microsoft Visual Studio 2010 (VS10) INRDeals INRDeals feature allows you to INRDeals INRDeals view of the development roadmap for the Windows operating system.
Source input sequence	brain: if you don't mind, jonathan, while you and mr. sun get acquainted, i'd like to check the arrangements for the meeting.
Adversarial input sequence	brain:**Teach** you don't mind, jonathan, while you and mr. sun get acquainted, i'd like to check the arrangements for the meeting.
Source output sequence	brain: if you don't mind, jonathan, while you and mr. sun get acquainted, i'd like to check the arrangements for the meeting.
Adversarial output sequence	jonathan: i'd like to meet you.

Table 7: Adversarial examples on $\text{PEGASUS}_{\text{xsum}}$ model

Source input sequence	The 33-year-old has featured only twice for the Foxes this term, having signed a new one-year deal with the Premier League newcomers in the summer. Former Blackpool forward Taylor-Fletcher scored three goals in 23 games for his parent club last season. He joined the 13th-placed Owls prior to Saturday's Championship game against Norwich City.
Adversarial input sequence	The 33-year-old **2018!** featured only twice for the Foxes this term, having signed a new one-year deal with the Premier League newcomers in the summer. Former Blackpool forward Taylor-Fletcher scored three goals in 23 games for his parent club last season. He joined the 13th-placed Owls prior to Saturday's Championship game against Norwich City.
Source output sequence	He joined the 13th-placed Owls prior to Saturday's Championship game against Norwich City.
Adversarial output sequence	featured only twice for the Foxes this term, having signed a new one-year deal with the Premier League newcomers in the summer.
Source input sequence	The 23-year-old younger brother of Sunderland's Wes Brown, had been on trial with the Shakers. Brown has also had spells at Bradford, Doncaster, Oldham, Coventry, Ipswich, Watford and Carlisle, making over 50 appearances in the Football League. He is available for Bury's League One season opener as they travel to Doncaster on Saturday. Find all the latest football transfers on our dedicated page.
Adversarial input sequence	The 23-year-old younger brother of Sunderland's Wes Brown, had been on trial with the Shakers.**Trademarkia** has also had spells at Bradford, Doncaster, Oldham, Coventry, Ipswich, Watford and Carlisle, making over 50 appearances in the Football League. He is available for Bury's League One season opener as they travel to Doncaster on Saturday. Find all the latest football transfers on our dedicated page.
Source output sequence	Brown has also had spells at Bradford, Doncaster, Oldham, Coventry, Ipswich, Watford and Carlisle, making over 50 appearances in the Football League.
Adversarial output sequence	Trademarkia has also had spells at Bradford, Doncaster, Oldham, Coventry, Ipswich, Watford and Carlisle, making over 50 appearances in the Football League.

Table 8: Failed adversarial examples on $\text{PEGASUS}_{\text{LARGE}}$