AUTOMATED SEARCH ENGINE PROMPT GENERATION FOR NAUTICAL CRIME

GOLNOUSH FARZANFARD[1], JACK KENDRICK[2], AND YOUSSEF MOUSAAID[3]

Abstract. Illegal and unreported fishing is a global problem that is estimated to contribute a loss of up to 26 million tonnes of fish annually, yet there is currently no extensive database of incidents worldwide. In this paper, we propose a dynamic system for the automatic generation of search engine prompts that yield reports of incidents related to illegal and unreported fishing. We also propose two metrics to evaluate the relevancy of a prompt's results to the topic at hand. By implementing an automated system of prompt generation and evaluation, large scale searches of the internet can be conducted autonomously, leading to the acquisition of larger amounts of relevant reports than would be possible with manual effort.

1. Introduction

Illegal and unreported fishing is a worldwide problem that contributes to the over-exploitation of fisheries, hinders fish population recovery, and is estimated to contribute a loss of up to 26 million tonnes of fish annually [2]. However, limited data about these crimes is available. Studies investigating fisheries crime have used datasets such as the Combined IUU Vessel List, containing 312 vessels, [10] and the Criminal Record of Fishing Vessels (CRFV) database [3],[4], containing 6583 incidents. The estimated scale of illegal and unreported fishing vastly exceeds the amount of data that is available to both researchers and law enforcement and thus it is necessary to increase the amount of reliable data that is available.

The CRFV database, which to our knowledge is the largest database of incidents related to fisheries crime, contains details of incidents and related offenses that took place between 2000 and 2020 and was compiled from four main sources: government reports (19%), third part reports (75%), automatic identification system track overlap analysis (5%), and confidential informants (0.3%) [4]. While the database is global in scope, due to the small number of incidents it contains, may not be fully representative of illegal and unreported fishing that goes undetected or unreported. Moreover, the incidents contained in the database were collected manually and thus building and maintaining the database, especially as time goes on, is a labor intensive and time consuming endeavour.

Motivated by this problem, Nautical Crime Investigation Services is developing a system to automatically identify crimes committed at sea from publicly available data. This system will generate search engine prompts, identify relevant results, and extract relevant information to form incident reports for illegal and unreported

[1]University of Lethbridge, [2]University of Washington, [3]University of Alberta.

fishing, as well as other crimes committed at sea. In this paper, we consider the question

> *How can we automatically generate prompts that yield relevant results?*

and propose a dynamic system for generating such prompts. Although we do concentrate on yielding results that are relevant, we also consider the diversity and bias of the results generated and propose future steps that will lead to a system yielding results that are to the topic at hand, diverse in content, and have minimal bias.

2. Method Overview

We propose a method of leveraging natural language processing techniques to generate keywords associated to unreported fishing catches and exploiting expected structure to generate a list of relevant search engine prompts. An overview of our pipeline can be seen in Figure 1

Based on experience with search engines and examining sample prompts that have yielded relevant results, we assume the following structure for all automatically generated prompts:

"Subject" + "Crime" + "Consequence" (+ "Specifics")

The subject of each prompt is expected to be some kind of watercraft and the crime will be some method of underreporting or misreporting a catch. The "Consequence" term may be simply that the watercraft was caught committing the crime or may be more specific, such as an arrest, fine, or trial. More specific information, such as breeds of fish or geographic areas, may be optionally included to generate more targeted search results.

We generate lists of keywords for each function in the prompt using a combination of keyword extraction and clustering techniques, see Section 3.1 for details. The pipeline is initialized with keywords that have been extracted and clustered from a list of high quality, manually generated search engine prompts that are known to yield results that are relevant to the topic at hand, reliable, and diverse in content. After the lists of keywords are generated, we randomly choose a keyword from each category and combine them to form a complete prompt that can then be fed into a search engine. Once a prompt has been entered into a search engine, we evaluate the prompt's performance based on the relevancy of the yielded results. In Section 3.2 we detail two metrics we have designed to quantify a result's relevancy, and how these metrics are used to quantify a prompt's overall performance. If the prompt does not perform well, it is discarded and a new prompt is generated.

As our prompts are passed through a search engine and relevant results are found, we generate new keywords by extracting relevant keywords. These keywords are assigned to a function within the prompt based on their similarity to words that already exist within the keyword list. If a keyword is not judged to be similar enough to the pre-existing keywords, it is discarded. By adding new words to keyword lists based on the already existing keywords, we eliminate the need to run a computationally expensive clustering algorithm with every iteration of the prompt generation process. However, it is possible that this process will discard some relevant keywords incorrectly. To combat this, we suggest storing all generated keywords, including those that have been discarded, and regularly reforming the keyword clusters by running the keyword clustering algorithms. As the lists of keywords get longer the clustering process becomes more complex but will ensure that all relevant keywords are considered and will allow for the generation of a more diverse set of prompts.

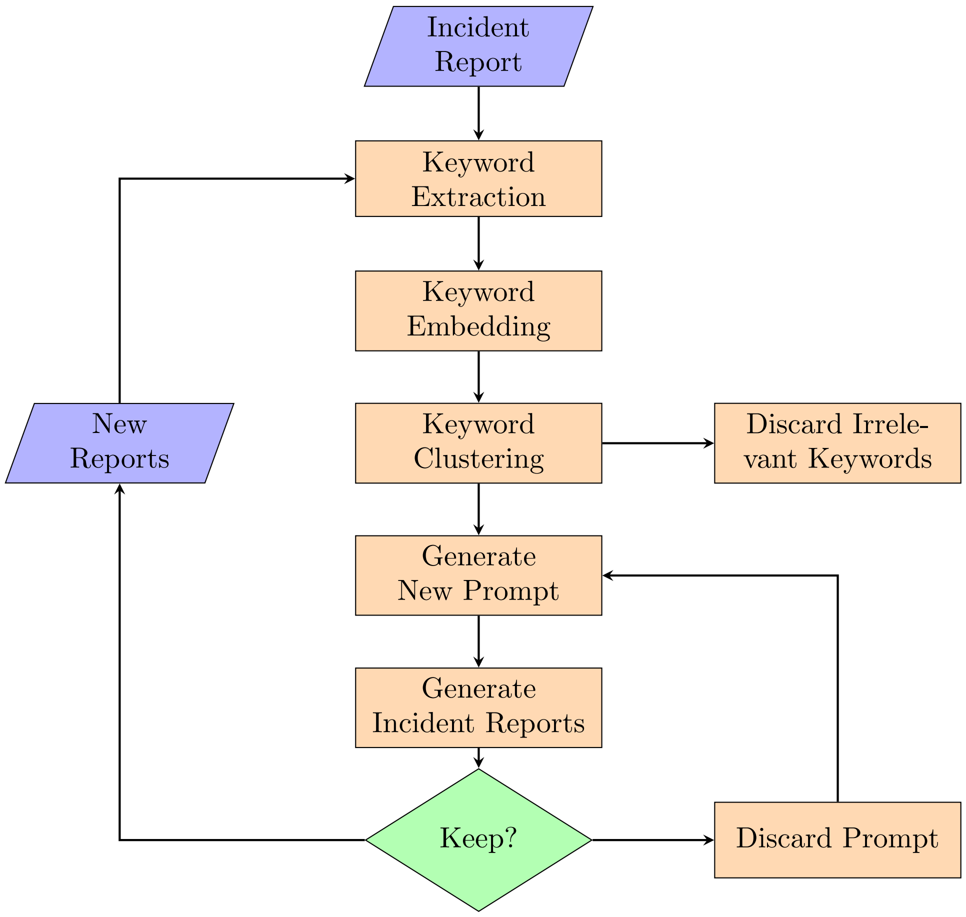

FIGURE 1. Proposed pipeline for dynamic prompt generation

3. Technical Details

In this section, we provide the technical details for the keyword extraction, keyword clustering, and performance evaluation steps in our prompt generation pipeline. First, we give a brief overview of some of the key natural language processing techniques that make this pipeline possible.

A key problem that arises from working with text data is that most algorithms are designed to work with numbers. For example, the K-Means and DBSCAN clustering algorithms take inputs as vectors and assign each vector to a cluster. In order to use these algorithms to cluster words, we must first embed each word as a vector. While it is possible for small datasets to assign words to vectors *ad hoc*, it is advantageous to use a word embedding that ensures the semantic meaning of any given word is preserved in the embedding process. In order for clustering algorithms to work well, it is important that words with similar meaning be "close" in the embedded space with respect to some metric such as Euclidean distance or cosine similarity. For example, for our purposes, we should have that the words "vessel" and "boat" are close together since these two words perform the same function in a search engine prompt, but "vessel" and "fishing catch" are far apart in the embedded space since they perform different functions within our prompt.

Some popular word embeddings, such as BERT [8] and GloVe [11] are language models that are pretrained on a large corpus of text data. These pretrained models can then be used to embed new words into a vector space while preserving the meaning of the word. Other word embedding methods, such as Total Frequence-Inverse Document Frequency (TF-IDF), are not pretrained and embed words based on their appearances in a given piece of text. In our model, we use BERT to embed keywords since the semantic meaning of keywords, rather than the frequency at which keywords appear, is most important to the task at hand.

Another key tool in natural language processing is the transformer deep learning architecture. The transformer architecture is based on the multi-head attention mechanism first introduced in the seminal paper [12] and has revolutionized the world of deep learning. Although the details of the transformer architecture are outside the scope of this paper, we note that it has had much success in tasks such as scoring text similarity and named entity recognition, which we use to evaluate prompt performance.

3.1. Keyword Extraction and Clustering.

To extract keywords from a given piece of text, we use the keyword extractor YAKE [5][6][7]. YAKE is a lightweight, open-source, unsupervised approach to keyword extraction. The unsupervised nature of the YAKE algorithm is beneficial for our purposes because we do not have a large corpus of machine readable, annotated documents that can be used to train a supervised model to extract the relevant keywords from a piece of text. Moreover, YAKE has the ability to extract key *phrases*, rather than just keywords, and does not have a threshold for the how often a word must appear in a piece of text to be considered a keyword. This means that we can identify "illegal fishing" as one term for our keyword list, whereas other algorithms may only identify "illegal" and "fishing" separately. Also, since the YAKE algorithm is term frequency-free, it performs well when extracting keywords from short pieces of text where each word appears exactly once, such as the manually generated search engine prompts used to initialize our pipeline. Anecdotally, we found YAKE to be a reliable, user friendly, and adaptable keyword extractor that performed well for our purposes.

Once keywords have been extracted, we use a combination of K-means and Density Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithms to assign keywords to a given function within our prompt. First, we used BERT to embed each keyword as a vector so that the numerical K-means and DBSCAN could be applied. Given a set of datapoints $x_1, \ldots, x_n \in \mathbb{R}^d$, and a number of clusters $k \geq 1$, the K-means clustering algorithm aims to assign datapoints to one of k clusters such that the variance within each cluster is minimized. Formally, the problem is to find a partition $C^* = \{C_1^*, \ldots, C_k^*\}$ of the points $x_1, \ldots, x_n$ such that for a specified metric d,

$$S^* \in \text{argmin}_C \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)$$

where each $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ is the center of the cluster C_i. The standard algorithm for solving the K-means problem is given in Algorithm 3.1.

Algorithm 3.1

```
Input: Set of points $x_1, \ldots, x_n \in^d$, number of clusters $k$
Output:
Initialize: Choose $k$ centroids $\mu_1, \ldots, \mu_k$ (e.g. via random partition)
$i \leftarrow 0$
converged ← False
while not converged do
    for each point $x_j$ do
        Assign $x_j$ to cluster $C_\ell^{i+1}$ if $\mu_\ell = \text{argmin}_\ell d(x_i, \mu_\ell)$
    end for
    for each cluster $C_\ell^{i+1}$ do
        Calculate new centroid $\mu_\ell^{i+1} = \frac{1}{|C_\ell^{i+1}|} \sum_{x \in C_\ell^{i+1}} x$
    end for
    $i \leftarrow i + 1$
    if for all $j, C_j^i = C_j^{i-1}$ then
        converged ← True

        Return Optimized clusters $C_1^i, \ldots, C_k^i$
```

Since we have four specified functions within our prompt structure, we use the K-means algorithm with $k = 4$ to cluster the embedded keyword vectors into each function. One restriction to the K-means algorithm is that it does assign every input data point to a cluster and so the output clusters may contain irrelevant keywords due to noise from the keyword extraction process. In order to identify and discard irrelevant keywords, we cluster keywords a second time using the DBSCAN algorithm.

The DBSCAN algorithm was first proposed in [9] and assigns data points $x_1, \ldots, x_n$ to clusters based on density with respect to some metric d. The algorithm requires two hyperparameters, M and ε, and classifies data points as one of four types: *core points, directly reachable, reachable* and *outliers.* A point x is considered a *core point* if at least M data points are within ε distance of x with respect to d. A point y is *directly reachable* if there exists some core point x such that $d(x, y) < \varepsilon$ and *reachable* if there exists a sequence $y_1, \dots, y_m$ of points such that y_1 is a core point, $d(y_m, y) < \varepsilon$ and $d(y_{i+1}, y_i) < \varepsilon$ for all $i = 1, \dots, m-1$. Given two points x, y, if there exists a third point z such that x and y are both reachable from z, then x and y are *density-connected.* Then, a cluster is a set of points such that

(1) Every pair of points x, y in the cluster is density connected
(2) Any point that is reachable from some point in the cluster is itself in the cluster.

If a data point is not a core point or (directly) reachable, it is considered to be an outlier and not assigned to any cluster. The precise details of the algorithm are given in [9].

The combination of K-means and DBSCAN clustering allows us to assign extracted keywords to clusters based on semantic meaning while removing any irrelevant keywords. However, it is not yet possible to automatically assign each of the four K-means clusters to the relevant function within the prompt. While it would be beneficial to develop a system that can automate this process, we note that, provided hyperparameters and word embeddings are chosen appropriately, a user need only review a handful of keywords from each cluster to assign it to the relevant function in the prompt structure.

3.2. **Performance Metrics.** Once a prompt has been generated and fed into a search engine, we scrape the text of each of the top 5 search results and evaluate the relevancy of each result using two metrics: *Similarity Score* and *NER Score.*

Our first metric, *Similarity Score*, uses a pre-trained transformer model to quantify the similarity of each sentence in a result to a user specified topic sentence. In our pipeline, we use the topic sentence “Vessel caught underreporting or misreporting fishing catch” as each prompt we generate should return results relevant to this topic. Note that the topic sentence should be predefined and should not be the same as the search engine prompt: evaluating the similarity of a result’s content to the prompt that generated it measures the quality of the search engine, not the quality of the prompt. The similarity of each result is given as a value between 0 and 1, with 1 indicating that the result and topic sentence are identical in meaning. After evaluating the similarity score of each of the top 5 search engine results, the similarity score of the prompt is defined to be the average similarity score over each of the results.

Our second proposed metric, *NER Score*, uses a second pre-trained transformer model that is specialized to identify named entities within a piece of text. When searching for reports of illegal and unreported fishing, there are several variables of interest for each incident. For example, we would like to be able to identify the vessel name, ownership, and flag, as well as information on what crime was committed and whether any arrest was made. We propose that each of these variables be treated as a named entity that can be found by a specialized NER model. Each variable of interest is assigned a weight based on its relevant importance and the NER score of each result is defined to be the weighted average of each of the variables of interest it contains. After evaluating the NER score of each of the top 5 search engine results, the overall NER score for a prompt is defined to be the average NER score across the top 5 results.

Once a prompt's similarity and NER scores have been calculated, the overall score given to a prompt is a weighted average of the two. The weight assigned to each score is user defined and should be fine tuned by manually evaluating the quality of a prompt's search results and comparing the perceived quality to the quality as determined by the similarity and NER metrics.

4. Results

We created initial clusters of keywords by extracting keywords from a list of 100 manually generated search engine prompts that are known to yield relevant and diverse results relating to unreported and illegal fishing. Once the keywords were extracted, we clustered keywords using K-means clustering, with $k = 4$ since there are four functions within our specified prompt structure, and eliminated noise using the DBSCAN clustering algorithm. We then generated a list of prompts by randomly choosing a word from each cluster to fill each function within the prompt. After generating the list of prompts, we retrieved results from Google News using SerpApi [1] and scraped all text from the top 5 search results for each prompt. The text of each search result was evaluated using both the similarity and NER metrics described in Section 3.2 and an overall similarity and NER score for each prompt was calculated by taking the average over the top 5 search results. The overall score was calculated as

$$\text{Overall Score} = 0.6 \times \text{Similarity Score} + 0.4 \times \text{NER Score}$$

We placed a higher weight on the similarity score than the NER score because there is currently no available NER model that can correctly identify our variables of interest. Fine tuning the pretrained models used in our pipeline requires a large amount of compute power, data, and time, which was unfortunately not available to us. Therefore, we were restricted to working with out-of-the-box pretrained models that could not quite perform the task at hand. This problem is particularly prevalent in the NER task. For example, consider the following excerpt from a sample incident report:

> The South Korean-flagged trawler belonged to the fleet operated by the Sajo Oyang corporation, notorious for its record of high seas transgressions, as documented by The Guardian. In recent years, the Oyang 77 had gotten in trouble in New Zealand for illegally dumping dead fish overboard, underreporting catch and failing to pay workers, according to a report from Oceana, a nonprofit focused on ocean conservancy. In February 2019, the Argentine Coast Guard discovered the trawler with its nets extended inside the EEZ. They

found more than 310,000 pounds of seafood on board. Leaving nothing to chance, they deployed a helicopter and an airplane to assist the Coast Guard in escorting the Oyang 77 to shore, releasing it after confiscating its fishing equipment and extracting a fine of 25 million Argentine pesos, or about $550,000.

The out-of-the-box NER model used in our pipeline identifies entities as follows:

```
South Korean | NORP | Nationalities or religious or political
groups
Sajo Oyang | PERSON | People, including fictional
Guardian | ORG | Companies, agencies, institutions, etc.
recent years | DATE | Absolute or relative dates or periods
Oyang | ORG | Companies, agencies, institutions, etc.
New Zealand | GPE | Countries, cities, states
Oceana | ORG | Companies, agencies, institutions, etc.
February 2019 | DATE | Absolute or relative dates or periods
the Argentine Coast Guard | ORG | Companies, agencies, institutions,
etc.
EEZ | ORG | Companies, agencies, institutions, etc.
more than 310,000 pounds | QUANTITY | Measurements, as of
weight or distance
the Coast Guard | ORG | Companies, agencies, institutions,
etc.
Oyang | ORG | Companies, agencies, institutions, etc.
25 million | CARDINAL | Numerals that do not fall under another
type
Argentine | NORP | Nationalities or religious or political
groups
about $550,000 | MONEY | Monetary values, including unit
```

While the model does a good job at identifying named entities in general, the identification is not optimal for our usage. For example, the model can identify that a South Korean ship is being discussed, but does not identfiy the ship's name and mistakenly identifies the company Sajo Oyang, who own the ship, as a person. Moreover, organizations such as The Guardian are recognized even though they are irrelevant to the incident being reported. Given enough data, time, and computational power, it is possible to fine tune the out-of-the-box model and this is something we hope to explore in the future, see Section 5.

Table 1 contains a list of sample prompts generated with our pipeline and their accompanying similarity, NER, and overall scores.

5. Conclusion and Future Work

We propose a system to automatically generate search engine prompts that yield results relevant to incidents of unreported and illegal fishing. Our pipeline leverages natural language processing techniques such as word embeddings, transformer

Prompt	**Similarity**	**NER**	**Overall**
Vessel caught undeclared catch.	0.4675	1.2800	0.7925
Vessel caught misreporting species.	0.5384	1.1600	0.7871
Vessel caught reporting species.	0.4451	1.2600	0.7711
Vessel caught falsifying catch.	0.4965	1.1600	0.7619
Vessel caught misreporting quota.	0.5368	1.0600	0.7461
Vessel caught misreporting fishing.	0.5711	1.0000	0.7427
Vessel caught falsified quota.	0.5563	1.0200	0.7418
Vessel caught falsified catch.	0.4671	1.1200	0.7283
Vessel caught underreporting volumes.	0.5048	1.0600	0.7269
Vessel caught misreporting quotas.	0.5302	1.0200	0.7261
Vessel caught underreporting catch.	0.4918	1.0600	0.7191
Vessel caught hiding fishing.	0.3793	1.2000	0.7076
Vessel caught reporting species.	0.4341	1.1000	0.7005
Vessel caught evade catch.	0.3959	1.1400	0.6935
Vessel caught underreporting fishing.	0.4854	1.0000	0.6912
Vessel caught falsifying tuna.	0.3798	1.1200	0.6759
Vessel caught unreported catch.	0.4865	0.9400	0.6679
Vessel caught misreporting fish.	0.4879	0.8800	0.6447
Vessel caught misreporting fish.	0.4875	0.8800	0.6445
Vessel caught misreporting catch.	0.4105	0.8800	0.5983
Vessel caught misreporting numbers.	0.3998	0.8800	0.5919
Vessel caught bypassing catch.	0.3537	0.9000	0.5722
Vessel caught reporting catch.	0.2518	1.0000	0.5511
Vessel caught reporting catch.	0.2518	0.9800	0.5431
Vessel caught fake catch.	0.1988	0.8800	0.4713
Vessel caught avoid catch.	0.2311	0.7800	0.4506

Table 1. List of sample prompts with accompanying Similarity and NER scores

models, and named entity recognition, to automatically generate lists of keywords related to our given topic and evaluate the performance of search engine prompts based on the results they yield.

While we have made some progress towards a system that can dynamically generate prompts to yield diverse, relevant results, there are many avenues that should be further explored to improve performance. The major issue with our current approach is that the out-of-the-box pretrained models used in both the keyword embedding and prompt evaluations processes are not fine-tuned to the nautical crime domain. This means that the keyword embedding and clustering process is not optimal and could lead to relevant keywords being discarded during DBSCAN clustering, or keywords being assigned to the wrong cluster. In order to dynamically generate prompts using a bank of keywords that grows as new reports of unreported and illegal fishing are retrieved, it is vital that relevant keywords are extracted reliably and embedded such that they can be assigned to the correct function within our prompt structure. Moreover, as discussed in Section 3.2, having a fine-tuned model is crucial for our proposed NER score. Our hope is that a fine-tuned model would enable our pipeline to correctly and reliably identify information such as the name of a vessel, the vessel's ownership, as well as details of the crime that has been committed. The identification of these variables of interest not only will allow for a more meaningful evaluation of a prompt's performance, but will also aid in the generation of nautical crime incident reports that could then be added to a fisheries crime database.

Currently, we generate prompts by randomly choosing a term from each cluster of keywords, but this can lead to prompts being repeated and does not ensure that the optimal set out of prompts is generated. Once fine-tuned models are available for the nautical crime domain and the correct hyperparameters for our proposed metrics have been determined, an interesting direction to explore would be the creation of a model such as a neural network that can be trained to generate an optimal set of prompts.

While we have concentrated on generating prompts that yield results relevant to the topic of unreported and illegal fishing, it is also important to generate results that are diverse and have minimal bias. One way to encourage the retrieval of diverse results is by retrieving search results globally rather than from only one location. In the future, this can be achieved using SerpApi to specify the location from which results should be retrieved. Additionally, English language bias can be diminished by using machine translation to translate prompts from English into other languages. The translated prompts can then be given to a search engine and the results can be evaluated.

Although it is impossible to truly eliminate bias, since we cannot control whether illegal and unreported fishing goes undiscovered and unreported, it is important that our system generates prompts that yield all relevant results and do not focus on certain incidents or geographic areas while other go underrepresented. Unreported and illegal fishing is a global problem that has economic and environmental impacts worldwide: it is important to ensure that the results yielded by our automatic prompt generation pipeline reflect the diverse and global nature of the issue. By automating the prompt generation process, we hope that large scale, global internet searches can be carried out with minimal human labor to ensure that all available information is retrieved and documented.

6. Acknowledgements

The team would like to thank PIMS and the M2PI workshop organizers, in particular Dr Kristine Bauer and Ian Allison, for the opportunity to take part in this workshop. We are also incredibly grateful to and thankful for the guidance of our mentor Sogol Ghattan of Nautical Crime Investigation Services, without whom we would not have been able to explore this problem.

References

1. *SerpApi: Google Search API.*
2. David J. Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R. Beddington, and Tony J. Pitcher, *Estimating the Worldwide Extent of Illegal Fishing*, PLoS ONE **4** (2009), no. 2, e4570.
3. Dyhia Belhabib and Philippe Le Billon, *Tax havens are the tip of the iceberg*, Nature Ecology & Evolution **2** (2018), no. 11, 1679–1679.
4. ———, *Fish crimes in the global oceans*, Science Advances **8** (2022), no. 12.
5. Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt, *YAKE! Keyword extraction from single documents using multiple local features*, Information Sciences **509** (2020), 257–289.
6. Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt, *A Text Feature Based Automatic Keyword Extraction Method for Single Documents*, 2018, pp. 684–691.
7. ———, *YAKE! Collection-Independent Automatic Keyword Extractor*, 2018, pp. 806–810.
8. Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference **1** (2018), 4171–4186.
9. Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, (1996).
10. Victor Galaz, Beatrice Crona, Alice Dauriach, Jean-Baptiste Jouffray, Henrik Österblom, and Jan Fichtner, *Tax havens and global environmental degradation*, Nature Ecology & Evolution **2** (2018), no. 9, 1352–1357.
11. Jeffrey Pennington, Richard Socher, and Christopher D. Manning, *GloVe: Global vectors for word representation*, EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference (2014), 1532–1543.
12. Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention Is All You Need.*