

Flamingo: Environmental Impact Factor Matching for Life Cycle Assessment with Zero-Shot Machine Learning

BHARATHAN BALAJI, VENKATA SAI GARGEYA VUNNAVA, NINA DOMINGO, SHIKHAR GUPTA, HARSH GUPTA, and GEOFFREY GUEST, Amazon, USA
ARAVIND SRINIVASAN, University of Maryland and Amazon, USA

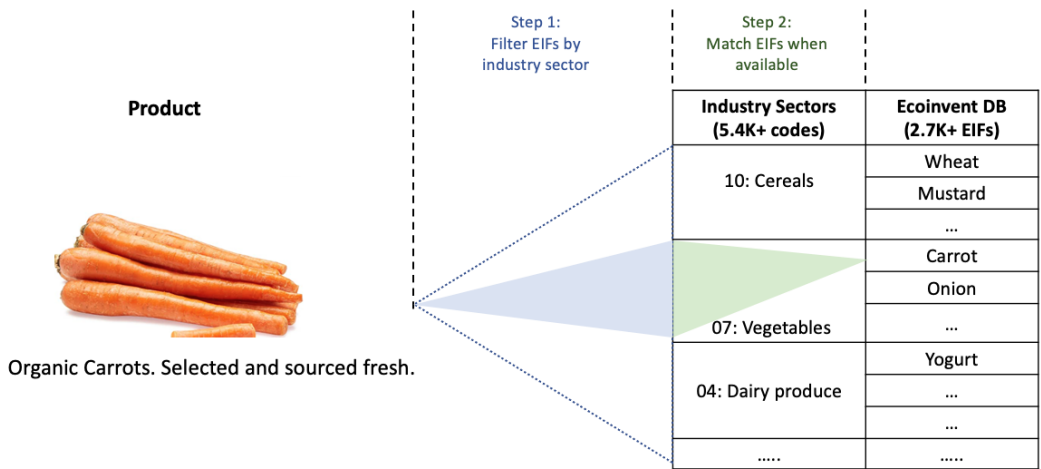


Fig. 1. Environmental impact factor (EIF) selection is a key aspect of product carbon footprinting. Flamingo automates selection of an EIF using pre-trained neural language models. It uses industry sector codes to identify when an EIF is not available in the database.

Consumer products contribute to more than 75% of global greenhouse gas (GHG) emissions, primarily through indirect contributions from the supply chain. Measurement of GHG emissions associated with products is a crucial step toward quantifying the impact of GHG emission abatement actions. Life cycle assessment (LCA), the scientific discipline for measuring GHG emissions, estimates the environmental impact associated with each stage of a product from raw material extraction to its disposal. Scaling LCA to millions of products is challenging as it requires extensive manual analysis by domain experts. To avoid repetitive analysis, environmental impact factors (EIF) of common materials and products are published for use by LCA experts. However, finding appropriate EIFs for even a single product under study can require hundreds of hours of manual work, especially for complex products. We present Flamingo, an algorithm that leverages natural language machine learning (ML) models to automatically identify an appropriate EIF given a text description. A

Authors' addresses: Bharathan Balaji, bhabalaj@amazon.com; Venkata Sai Gargeya Vunnavu, gvunnavu@amazon.com; Nina Domingo, nggdom@amazon.com; Shikhar Gupta, gupshik@amazon.com; Harsh Gupta, hrshgup@amazon.com; Geoffrey Guest, gmg@amazon.com, Amazon, Seattle, Washington, USA; Aravind Srinivasan, srinarav@amazon.com, University of Maryland and Amazon, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

2834-5533/2023/8-ART111 \$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

key challenge in automation is that EIF databases are incomplete. Flamingo uses industry sector classification as an intermediate layer to identify when there are no good matches in the database. On a dataset of 664 products, our method achieves an EIF matching precision of 75%.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**; **Information retrieval**; • **Computing methodologies** → **Natural language processing**; • **Applied computing**;

Additional Key Words and Phrases: environmental impact factor, life cycle assessment, carbon footprint, semantic matching, natural language processing, HS codes, Flamingo

ACM Reference Format:

Bharathan Balaji, Venkata Sai Gargeya Vunnava, Nina Domingo, Shikhar Gupta, Harsh Gupta, Geoffrey Guest, and Aravind Srinivasan. 2023. Flamingo: Environmental Impact Factor Matching for Life Cycle Assessment with Zero-Shot Machine Learning. *ACM J. Comput. Sustain. Soc.* 37, 4, Article 111 (August 2023), 24 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Greenhouse gas (GHG) emissions from human activities are warming the planet, and will lead to hazardous climate events in the absence of mitigation efforts [38]. While information on global and country-level emissions is generally available [1], there is a lack of granular emissions data that can inform mitigation. Life cycle assessment (LCA) is a standard environmental accounting method used to estimate higher granularity GHG emissions associated with an activity or a product. These emissions are often referred to as its carbon footprint, and are reported in terms of global-warming potential in units of mass of carbon dioxide equivalent (e.g., kgCO₂e) [17].

LCAs for consumer products are of special interest as they account for greater than three-fourths of global GHG emissions [32], thus, data-driven approaches that perform LCA at scale are essential to informing carbon-abatement strategies [26]. Moreover, LCA can be used to estimate other environmental impacts such as electricity used, toxic waste, etc. LCA estimates emissions in each stage of a product: raw material extraction, manufacturing, transportation, use, and disposal/reuse. The effort to acquire direct measurements for each aspect of a product can be prohibitively expensive [53], and therefore, domain experts use the outputs of existing LCA studies to estimate emissions of common materials, products, and activities associated with the life cycle of their subject [57]. For example, an LCA on the "production of a cotton t-shirt" might rely on the results of LCAs focused on "cotton production" or "transport by truck". We refer to the outputs of these reference LCA studies as *environment impact factors* (EIFs). Databases such as Ecoinvent [57] and GaBi [50] collate these EIFs for use by LCA practitioners.

There is a lack of automation tools that integrate with EIF datasets. Often, EIF datasets are shared as spreadsheets, and domain experts use string matching or explicit rules to match a product or activity to an appropriate EIF [11, 45]. It can take hundreds of staff hours to match EIFs for LCA of even a single product [33]. To mitigate the manual overhead, we propose using natural language processing (NLP) based machine learning (ML) methods to perform this matching. To our knowledge, we are the first to study use of neural ML based EIF matching. A simple approach is to use neural language models trained on web data for semantic textual similarity [42], where the EIFs can be matched based on the distance between the embedding of the query text and the EIF description. A major advantage of this approach is that no training data is required, as off-the-shelf pre-trained models can capture synonyms and conceptual relationships. We consider this as our baseline algorithm.

A key characteristic of the EIF matching problem is that the databases are incomplete. Many products such as mushrooms or socks do not have an EIF, because either no LCA study exists or the published LCAs have not been ingested into the database. LCA experts search across multiple

databases for a match, and use an approximate value with higher uncertainty when no match exists, e.g., average EIF of vegetables as a proxy for red bell peppers [11]. However, pre-trained ML models are not trained to identify when an appropriate match does not exist.

We present a novel zero-shot ML algorithm, called Flamingo¹, that introduces an intermediate classification layer in semantic search to identify when an EIF is missing, and improves the performance of EIF matches. Flamingo classifies the input query text to a standard industry code, and uses semantic text matching to identify the closest EIFs within the industry code. When there are no EIFs available for an industry code, Flamingo predicts that no appropriate match exists. We use an industry sector classification called the Harmonized System (HS) code [9], which is specifically designed for categorizing products based on their material composition and manufacturing complexity. HS codes are hierarchically organized, are used globally for import/export taxes, and are refreshed by the World Customs Organization every 5 years [58]. Flamingo exploits the HS code hierarchy to navigate the precision versus recall trade-off associated with an EIF match.

We evaluate Flamingo on a dataset of 664 products from an e-commerce retailer. We use annotations from crowd workers to identify if an EIF predicted by Flamingo is an appropriate match, or if no match exists. Our results show that Flamingo matches EIFs to products with a precision of 75%, and outperforms the semantic text similarity baseline by 8.4%. We open-source our code with a permissive license².

The rest of the paper is structured as follows. In section 2, we briefly introduce LCA and summarize prior work in the literature on finding EIFs for LCA. We also summarize the relevant text-matching works in the literature and their applicability to finding appropriate EIFs. In section 3 we introduce our problem statement. In section 4, we describe the datasets used in our experiments. In section 5, we describe the proposed methods and provide illustrative examples. In section 6 we describe the process for collecting ground truth data. In section 7 we summarize the results of our experiments. Finally, in section 8, we close with thoughts for future work.

2 BACKGROUND AND RELATED WORK

In this section, we briefly explain how LCA is used for carbon footprint estimation, and the role of EIFs in performing LCAs. We summarize the prior works on automation of matching EIFs for a given query in the LCA literature, and the key challenges that remain unaddressed. We then summarize the relevant text-matching literature, along with its applicability to the EIF matching problem.

2.1 Life cycle assessment

LCA was introduced as a systematic method to compare product design choices in terms of energy use, waste, and other environmental impacts [19, 23]. It has since been adopted as a standard method for carbon footprinting by both the International Standards Organization [16] and the GHG Protocol [51]. LCA can be categorized into two types: Economic Input-Output LCA (EIO-LCA) and Process-LCA. EIO-LCA uses transactions across industries in an economy to obtain an approximate impact assessment at an industry sector level [60]. This type of LCA is associated with aggregation issues, as for example, different types of paper products are assumed to have the same GHG impact per unit of sale price regardless of how they were manufactured. Process-LCA, on the other hand, produce higher granularity carbon footprints through detailed tracking of emissions from each life-cycle stage of a product. Figure 2 shows a Process-LCA of a paper towel as an example, which requires accounting of how the pulp was sourced, how it was processed across multiple suppliers,

¹Flamingos use their beaks to filter out algae and crustaceans from their food; our algorithm filters out unrelated EIFs.

²Open source code repository – <https://github.com/amazon-science/carbon-assessment-with-ml>

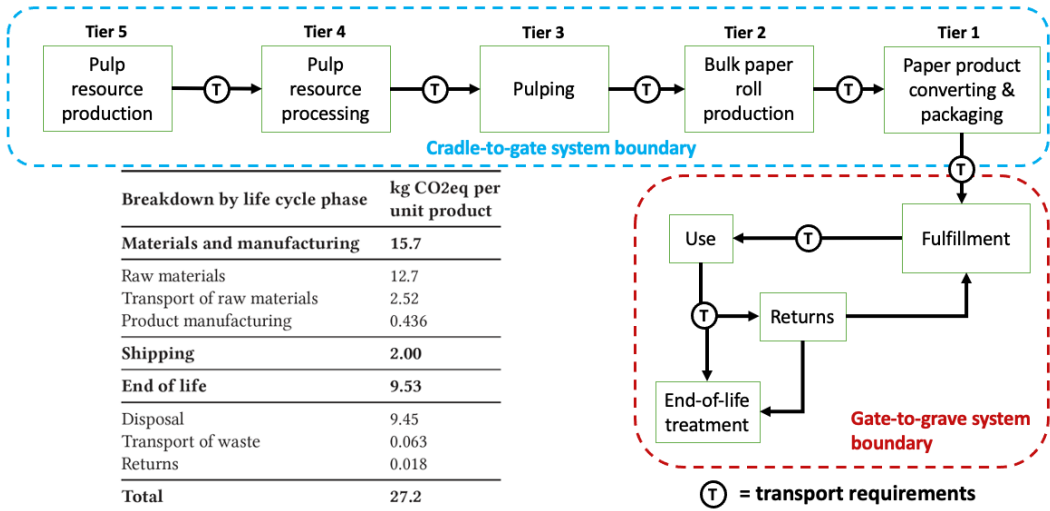


Fig. 2. Life cycle assessment of a paper towel made with 75% recycled materials [24].

how materials were transported between stakeholders, what percentage of products sold was returned, and whether the paper roll was composted, landfilled, or recycled. As it is challenging, and sometimes infeasible, to collect direct emissions data in such high detail, LCA experts use EIFs published in prior studies as an estimate of the GHG emissions associated with a product, material, or activity for which they do not have direct measurements [33, 45]. Ecoinvent [57], GaBi [50], and AGRIBALYSE [12] are some of the common EIF databases used in the industry.

Finding the right EIF can be time consuming [33] as exact string match does not capture synonyms (e.g., milk and dairy, maize and corn), abbreviations (e.g., Ni-Cd and Nickel Cadmium), technical terms (e.g., sodium chloride is same as salt), or category relationships (e.g., basmati is a type of rice). A few solutions have been proposed to overcome this challenge in the literature [11, 33, 45]. Meinrenken et al. [33] propose a linear regression algorithm, where they categorize the query text to an industry sector, and use a regression based on price to estimate the GHG emissions. However, the categorization is done manually, all the EIFs within an industry sector are averaged together which increases the variance of the estimate, and heuristics are used to remove outliers. Clark et al. [11] matched ingredients to EIFs for food products. They also rely on a mapping of EIFs to pre-defined food categories (e.g., berries, cheese). In addition, they manually create search terms which are synonyms or sub-types of a given food category (e.g., pecorino is a cheese) so they can improve exact string matches. To reduce variance of emission estimates, they use a three-level hierarchical categorization so that specificity of a match increases when possible, e.g., use strawberry instead of berries. Our algorithm uses a similar hierarchical industry classification, but does not require manual specification of search terms. As a result, our solution scales to all EIFs in the database, and is not limited to food EIFs. In contrast to these prior works, Flamingo does not require manual steps or heuristics, and can match to specific EIFs in the database instead of industry sectors.

ML has been used to automate other aspects of LCA, as covered in a survey by Algren et al. [2]. Specific to EIF estimation, Sousa and Wallace [48] directly estimate carbon dioxide emissions based on product attributes to inform design decisions. They used neural network regression on carefully chosen product descriptors to estimate emissions. They train different models for high and low energy use products. The model was trained on EIFs of 53 products, and predictions on a test set

of six products gave an error of 40%. However, their product descriptor required details such as mass, percentage contribution from different type of materials (ceramics, fibers, metals, plastics, etc.), energy source and more. In contrast, we identify if an EIF in the database can be matched to a given text description and do not place any restrictions on the input. We also test our predictions on a much larger dataset of 664 products.

2.2 Text Matching

Text search algorithms were envisioned with the advent of digital computers [7, 46]. Much of the early search algorithms were based on exact string matching [6, 54], and is still prevalent in spreadsheets used for EIF searching by LCA experts. Exact matches work well for small strings, but have poor recall with long search queries. BM25 [43, 44] and TFIDF [41, 49] are probabilistic algorithms that overcome this challenge by weighting words with their relative frequency of occurrence. They support both long search queries and rank results based on a relevance score. BM25 is the default search algorithm in modern databases such as Elastic Search [18]. To our knowledge, BM25 has not been considered for EIF search in the literature, and we include it as a baseline algorithm in our evaluation. We do not include exact match as a baseline as it is challenging to identify the keyword to use for the query, and use of the entire product description leads to spurious matches with close to 0% precision.

Neural search algorithms improve on exact string match by learning semantics such as synonyms and contextual relationships [34]. However, they require a large dataset of queries with matched results for training the model [20, 22]. No such dataset exists for searching EIFs, and therefore, we focus on zero-shot methods that do not require training data. Neural language models such as BERT [28] and GPT [40] reduce the reliance on training data by using a self-supervised objective such as predicting the next word in a sentence. Use of the Transformer architecture enables training on web-scale datasets [56]. Sentence transformers, called SBERT, build on these works, and have emerged as a strong zero-shot algorithm for semantic text matching [42, 59]. They are trained on web text to create a vector representation (a.k.a embedding) of an input sentence. We can identify if two sentences are similar by measuring the distance between their embeddings. Using SBERT, we can find the closest EIF that matches a query text by identifying the EIF text embedding that is closest to the query embedding. Balaji et al. [3] use SBERT to find EIFs for EIO-LCA, while we focus on finding EIFs for Process-LCA. Unlike Process-LCA EIF databases, EIO-LCA EIF databases are complete by definition as the EIF corresponds to industry sectors defined by the national governments. Therefore, Balaji et al. [3] do not address the problem of identifying when no EIF matches exist in the database. We include SBERT as a baseline in our evaluation.

One of the challenges with a neural search solution is that they are not designed to identify when an appropriate match does not exist [13, 52]. A simple method is to threshold the distance between embeddings beyond which an EIF is not an appropriate match [63]. However, as we show in our evaluation, a threshold based solution leads to incorrect predictions due to distance miscalibration. The SBERT model can be fine-tuned on a search dataset so that the distances are calibrated [52]. Another method is to train a separate model that determines if the predicted match is correct or incorrect [21]. But these solutions require labeled training data, and do not meet our zero-shot objective. We propose a novel solution, where we first classify the EIFs and the query text to an industry sector. If the industry sector of the query does not match with any entry in the EIF database, we can predict that there is no suitable match for this query.

We use the HS codes for our industry sector classification, which is hierarchically organized from 2 digits to up to 12 digits, where the level of hierarchy is indicated with a 2-digit increment [9]. Figure 3 shows an example. A major advantage of HS codes is that they are already used for import/export taxation worldwide, and therefore, datasets that map products to HS codes are

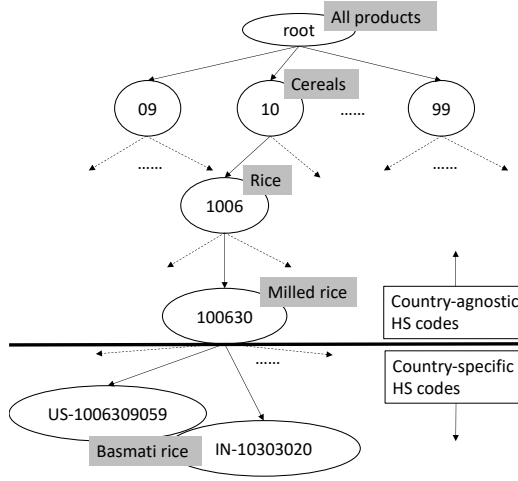


Fig. 3. An example of HS codes, along with its hierarchical structure.

readily available. We use up to 6 digits of the HS code (HS6), which translates to $\sim 5.4\text{K}$ unique codes. We leverage an existing dataset of products mapped to HS6 codes to learn a supervised classifier that can predict the HS code for any text input.

Prior works have proposed ML approaches to automate HS code classification as it is used to determine customs tax rate [10, 14, 15]. Chen et al. [10] consider a neural machine translation approach with a hierarchical loss function, and obtain an accuracy of 85% on a dataset of $\sim 1\text{M}$ records. Lee et al. [31] use a supervised, hierarchical sentence retrieval approach to classify 129K electrical products with an accuracy of 89.6%. Du et al. [15] use two parallel neural networks in a Siamese architecture for classifying HS codes, one network uses a hierarchical sequence of LSTM modules on word vectors of text input, and another uses graph attention network based on word co-occurrence. On a dataset of $\sim 400\text{K}$ samples, they obtain an average accuracy of 85%. We treat HS code classification problem as an example of extreme label classification [4, 5]. We use the PECOS algorithm, which leverages the label hierarchy and semantic similarity of labels to improve classification performance [61]. Flamingo is agnostic to the method used to classify HS codes, we use PECOS in our experiments as it achieves state-of-the-art in multiple extreme label classification benchmarks. On our dataset of 746K products, we obtain a classification accuracy of 82% which is commensurate to the results reported in prior works. We also use SBERT to identify the best matching HS codes for a given text description with a zero-shot approach. To our knowledge, zero-shot methods to classify HS codes have not been studied in the literature.

3 PROBLEM STATEMENT

Given a query text $q_i \in \mathcal{Q}$, our objective is to find an appropriate EIF e_i from a given set of EIFs \mathcal{E} . The query text can be a product description or a specific aspect of a product that an LCA expert will attach an EIF to such as the material a product is made of. All EIF datasets include text metadata that describes its characteristics, and we assume the text is available for query matching. It is possible that there is no appropriate EIF available for a given query, and the algorithm should output \emptyset , i.e. ‘no match’, for such cases.

We assume there is no training dataset available that matches query text to EIFs. It is difficult to obtain high-quality annotations for a dataset that is large enough to train models which generalize

Table 1. Summary of datasets used in the paper

| Type | Size | # Match | # No Match | # No clear label | # Used |
|------------------------|-------|-------------------------------|------------|------------------|--------|
| Product to EIF (small) | 100 | 22 | 68 | 10 | 90 |
| Product to EIF (large) | 967 | 277 | 387 | 303 | 664 |
| Food ingredient to EIF | 272 | 87 | 177 | 8 | 264 |
| Product to HS code | 746K | No filter | | | 746K |
| HS code descriptions | 6709 | Filter for HS6 codes | | | 5388 |
| Ecoinvent EIFs | 19128 | Filter for reference products | | | 2770 |

to all queries and EIFs. Although we do have a small dataset, we only use it for validation of methods. Even for few-shot methods, we need to have a few labeled examples per EIF, which are in the thousands. It is possible to use EIF text descriptions to reduce the reliance on labeled examples for every class [64], we leave exploration of such methods for future work.

4 DATASET

In this section, we describe the dataset we use to evaluate the performance of Flamingo in matching a query text to an EIF. Table 1 summarizes the datasets used in the paper.

The dataset—which we refer to as D_Q —is a set of 967 products sold by an e-commerce retailer. It includes a variety of products such as ice makers, electric fans, toasters, adhesives, etc. Given a product, we concatenate its title, description, and additional attributes into a single string as input to the classifier. For each product, the dataset includes the ground truth HS6 code as well as a matching EIF from the Ecoinvent dataset [57], if a match exists. We use the 2017 version of HS codes, and EIFs from the Ecoinvent v3.7. Human annotations are used to validate the EIF matches in the dataset: the annotation process is described in Section 6. All the product descriptions are in English—while the text-similarity models we use can generalize to any language, we restrict the language because our annotation team only understands English fluently. We annotate a subset of 100 products by LCA experts to evaluate the quality of annotations by non-experts. To evaluate generalization beyond products, we introduce another dataset of 272 food ingredients that are matched to EIFs with annotations.

Ecoinvent EIFs include metadata such as impact factor name, reference product, units, data quality, valid years, geographic specificity, and industry classification. We use the attribute ‘reference product’ as the basis for matching EIFs because it provides a non-technical but precise description of the EIF (e.g., wheat, yogurt). Once the EIFs with the reference product are identified, it is easy to add rules to increase specificity by additional attributes such as location. Ecoinvent contains 19K EIFs, but only 3.2K unique reference products. The dataset includes EIFs that are not related to carbon footprint of consumer products such as those related to construction, operation of equipment or transportation of goods. We filter these out to get 2.7K unique EIFs.

We refer to an individual HS code as $h_i^\delta \in \mathcal{H}$, where δ refers to the number of digits in the code. We obtain the text description of HS6 codes from <https://unstats.un.org/>. We use a dataset that maps 746K products to their corresponding HS6 codes for learning a supervised model to predict HS6 code given text as input: we refer to this dataset as D_H . The dataset consists of a variety of products (e.g., shoes, watch, coat), and comprises 2.5K unique HS6 codes. Note that this is a subset of the full set of HS6 codes, $|\mathcal{H}| = 5400$. The distribution of the number of products per HS6 code is skewed, with 10% of HS6 codes accounting for 86% of the products. Despite the skew and

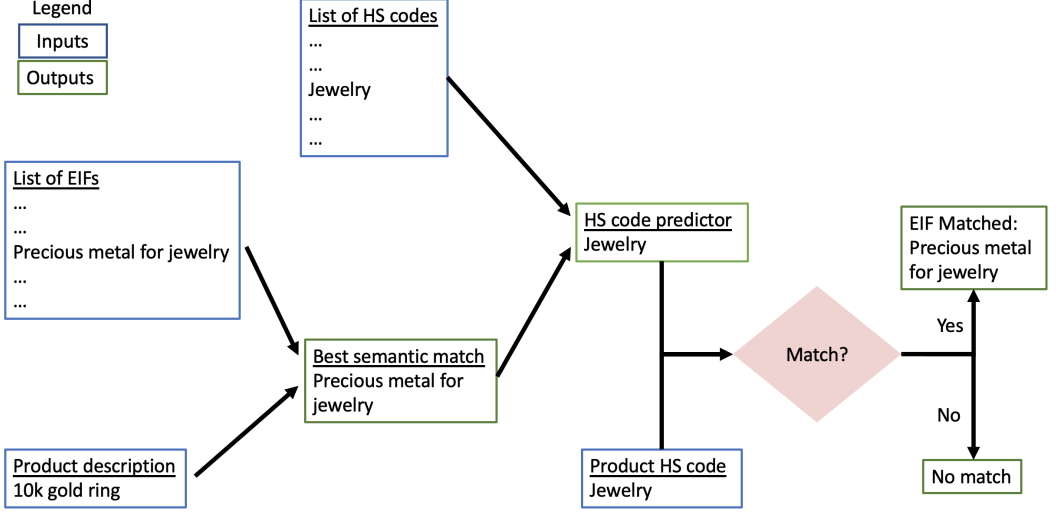


Fig. 4. Flowchart describing the Flamingo algorithm with a ‘10k gold ring’ as an example query. The matching EIF is found by semantic similarity based search on the EIF dataset. The HS code predictor is used to identify the HS code for the matched EIF. HS code predictor uses either semantic text similarity (FlamingoZero) or supervised PECOS algorithm (FlamingoPECOS). If the HS code of the EIF and the given query match, the final prediction is EIF, otherwise ‘no match’.

reduced cardinality, the HS6 codes in D_H contain a representative set of products, and we expect a large overlap with a generic product query.

5 METHODS

In this section, we describe the Flamingo algorithm. Our method relies on a zero-shot matching algorithm called SBERT [42] to map a query text to an EIF and introduces an intermediate industry classification code layer to identify when no good match exists. We summarize the notations used in Table 2.

Given a query text q_i , our objective is to both predict if a matching EIF exists in the database, and retrieve the matching EIF e_i when one exists. Critically, we do not rely on a supervised dataset of query to EIFs commonly used in prior works [21, 52]. Zero-shot matching algorithms, such as SBERT [42], can identify the best matching EIF but are inadequate at identifying when the EIF is not a good match based on a distance threshold. We improve on SBERT using industry sector classification as defined through HS codes [9]. We only consider the first 6 digits of the HS codes, called HS6, as they are globally applicable. Higher resolution HS codes with 8 or more digits are country-specific, and our methods can be extended to these if needed. Figure 4 provides an illustration of the Flamingo algorithm with an example.

We use a model M_H that predicts the HS6 code for a given text input τ_i . Given a HS6 code, we can lookup the corresponding HS4 and HS2 codes from the code hierarchy. Therefore, we have: $h_i^\delta = M_H(\tau_i, \mathcal{H})$. We propose two types of HS6 classifiers. The first is a zero-shot method that leverages the text description of HS6 codes and finds the best match using SBERT: we refer to the resulting method as **FlamingoZero** or M_H^Z . The second is a supervised classification method that leverages the product to HS6 dataset D_H using the PECOS algorithm [61]: we refer to this method as **FlamingoPecos** or M_H^P . While we expect the supervised model M_H^P to outperform the zero-shot

Table 2. Summary of notations used throughout the paper

| Notation | Description | Notation | Description |
|---------------|----------------------------------|----------|----------------------------------|
| D_Q | Query to EIF dataset | q_i | Query i |
| h_i^δ | HS code i with δ digits | e_i | EIF i |
| \mathcal{H} | Set of all HS codes | M_H | Model to predict HS6 code |
| τ_i | Text input i to match HS code | M_H^Z | SBERT model to predict HS6 code |
| D_H | Product to HS code dataset | M_E | Model to match EIF given a query |
| \mathcal{E} | Set of all EIFs | M_H^P | PECOS model to predict HS6 code |

model M_H^Z in the test split of the D_H dataset, M_H^P can under-perform when predicting the HS6 codes for EIFs due to domain shift. For example, chemical EIFs such as ‘1,1-dimethylcyclopentane’ are not available in the labeled HS6 dataset D_H , and will be mispredicted by the supervised model. As zero-shot methods are unbiased, they do not suffer from such a domain shift but their performance may suffer due to lack of any supervision. We study the trade-offs between the two approaches in our evaluation (Section 7).

Algorithm 1: Pseudocode of Flamingo

Input : $q_i, \mathcal{E}, \mathcal{H}, \text{cosine_threshold}$
Output: e_i
 $h_q^\delta = M_H(q_i, \mathcal{H})$
 $e_i, \text{cosine_score} = M_E(q_i, \mathcal{E})$
 $h_e^\delta = M_H(e_i, \mathcal{H})$
if $h_q^\delta \neq h_e^\delta$ **then**
 $e_i = \emptyset$
if $\text{cosine_score} < \text{cosine_threshold}$ **then**
 $e_i = \emptyset$
return e_i

Algorithm 1 details the steps in Flamingo. For a given query text q_i , we first predict its HS codes h_q^δ if not provided. Next, we find the best EIF e_i that matches the given query q_i using the SBERT model M_E . We then predict the corresponding HS code h_e^δ for the EIF e_i using the model M_H . If the HS codes of both the EIF and the query match, and the cosine similarity score is higher than the threshold, we output the best match EIF e_i as the final prediction. Otherwise, the output is ‘no match’ \emptyset . We include three variants of the algorithm, where we match EIFs to query based on their HS2, HS4, and HS6 codes respectively. Reducing the number of digits in the HS code helps reduce the specificity of the classification. Therefore, a higher-level HS code increases the precision of finding an appropriate EIF match at the expense of reducing the precision of predicting ‘no match’. We append the HS code we use to the name of algorithm to specify the variants, e.g., FlamingoZeroHS4.

Figure 5 shows examples of products being matched to EIFs based on text descriptions using Flamingo. The first example shows the importance of semantic matching, as the given product description does not mention ‘jewelry’ or ‘precious metal’. The EIF is considered a match as HS codes of the product and EIF are the same. In the second example, finding a match is easier with the keyword ‘coffee’. However, notice that the HS codes are slightly different – the product corresponds





| | Product Description | Product HS code | Best EIF by Semantic Matching | EIF HS code | Label |
|---|---|-----------------------|-------------------------------|---------------------|----------------------------|
|  | 10k Gold Imported Crystal March Birthstone Ring | Jewelry | Precious metal for jewelry | Jewelry | Precious metal for jewelry |
|  | Dark Roast Whole Bean Coffee | Coffee; roasted | Coffee, green bean | Coffee; not roasted | Coffee, green bean |
|  | Car tire brush | Brooms, brushes, mops | Vacuum cleaner | Vacuum cleaners | No match |
|  | Women's Belice Ballet Flat 100% synthetic | Footwear | Textile, non-woven polyester | Fabrics, woven | No match |

Fig. 5. Examples of product to EIF matching by Flamingo. The algorithm predicts that the given EIF is a match if the HS codes of the query and EIF match. The HS codes match for the first two examples, whereas they do not for the last two.

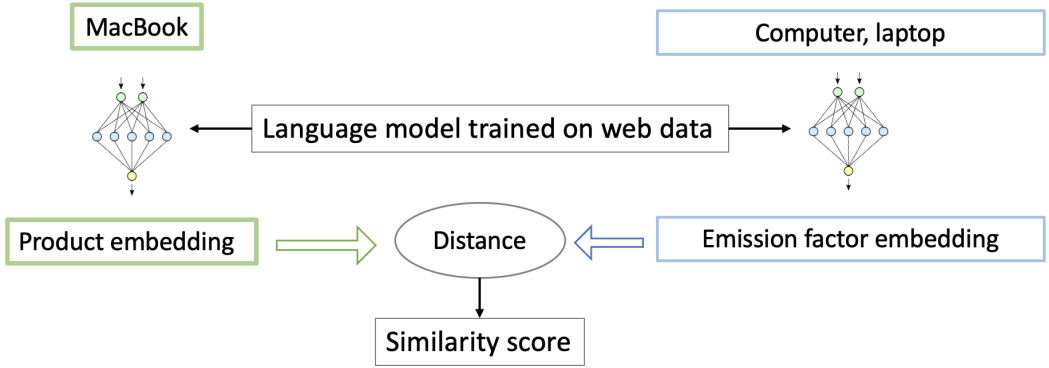


Fig. 6. An example of semantic similarity matching using a neural network model trained on web text data. The model takes text as input and outputs a vector representation, called embeddings, such that closely matched inputs will have mapped to embeddings with high cosine similarity.

to roasted coffee whereas the EIF is for green bean coffee not roasted. Our annotators consider these close enough, and therefore the EIF is considered a match. While 6-digit HS codes won't match in this case, the 4-digit HS codes corresponding to 'coffee' will match. For the last two examples, there are no good EIFs in the dataset. While the best EIF found do match the product semantically, they are not appropriate from a manufacturing perspective. The corresponding HS codes for the product and EIF reveal this difference and help predict 'no match'.

We use a pre-trained SBERT model to convert an input text to its corresponding embedding. We use the 'all-mpnet-base-v2' model from `sbert.net` as it has the highest performance in semantic text similarity benchmarks. The model uses the BERT Transformer architecture [28], and is trained on 160GB of text corpora using MPNet self-supervised objective [47]. BERT uses the masked language model objective for self-supervision, where the model predicts missing words in a given

sentence. Training on web-scale data with this objective, and fine-tuning on downstream natural language tasks led to state-of-the-art results at its time of release. MPNet improves on BERT, and permutes the input sentence in addition to masking. The model is further fine-tuned on 1B sentence pairs from a variety of NLP datasets using cosine similarity as the distance metric, and a symmetric cross entropy loss proposed by Radford et al. [39]. We use the default hyper-parameters throughout, where the input sentence is cutoff after 128 tokens (about 100 words). Figure 6 shows an illustrative example of semantic text similarity.

We use the same model to predict both the HS codes (M_H^Z), and to rank the best match EIFs (M_E). For example, to predict the EIF match for a given query text, the model will output the EIF embedding that is closest to the query embedding as measured through cosine similarity. It is possible that there is a ‘no match’ even after filtering out EIFs. We use a threshold on cosine similarity distance to further filter out unrelated EIFs, and show the impact of using both a conservative and an aggressive threshold on the algorithm performance.

5.1 FlamingoPecos

We train a classifier M_H^P that predicts the HS6 code given text as input using dataset D_H . We leave out 20% of the data as test set. We vectorize the product text using TFIDF [41], and the HS6 classes are one-hot encoded for classification. TFIDF converts each word in the text to a number proportional to relevance in the dataset, determined through the relative frequency of use compared to other words. We use the ‘XR-Linear’ option of the PECOS algorithm to train the classifier [61], which uses logistic regression and hierarchical clustering of labels for classification. We use the default hyper-parameters.

We briefly describe the PECOS algorithm: PECOS is a tree-based method for solving extreme multi-label classification problems. We focus on the XR-Linear variant of the algorithm, the linear classifiers can be replaced following the same framework.

The classification problem can be framed as follows: train a model that can generate a subset of labels from label space \mathcal{Y} which are relevant to an input \mathbf{x} . Given training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,2,\dots,N}$, we first construct label representations which is generated by normalizing the sum of its positive sample features. In particular, label i ’s representation is $\frac{\sum_{j \in S_i} \mathbf{x}_j}{\|\sum_{j \in S_i} \mathbf{x}_j\|}$, where S_i is the set of samples which has a positive label i . Once we obtain the label representation, we can construct the label tree using the hierarchical k-means algorithm [35]. The leaves of the tree are the final output labels, which in our case are HS codes.

For every given input \mathbf{x} , PECOS XR-linear tree will generate a subset of HS codes along with a score capturing how relevant that HS code is to the input query. More specifically, every node in the tree is associated with a simple linear multi-label classifier, which produces a score measuring the relevance between the child of this node and the input \mathbf{x} . The score for the i -th node in l -th layer is formulated as follows,

$$(\mathbf{x}, \mathcal{Y}_i^{(l)}) = \sigma(\mathbf{w}_i^{(l)} \cdot \mathbf{x}), \quad (1)$$

where σ denotes an activation function (e.g., sigmoid) and $\mathbf{w}_i^{(l)} \in \mathbb{R}^d$ denotes a sparse vector of weight parameters. This score accumulates along the path from the root to the leaves, similar to the computation of conditional probability if we view the score in each node as the conditional probability.

Training: Let the weight vector for a node j be \mathbf{w}_j and all the training samples be $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,2,\dots,N}$. For each input \mathbf{x}_i , we can find a path from root to leaf \mathbf{y}_i . When we train \mathbf{w}_j , we will include the sample \mathbf{x}_i as a positive if node j is in the path from root to \mathbf{y}_i , and as a negative if node j is not in the path from root to \mathbf{y}_i but the parent of node j is in the path from root to \mathbf{y}_i . Once we collect

all the positives P_j and negatives E_j for node j , we can train the weight for node j , by solving a binary classification sub-problem:

$$\mathbf{w}_j = \arg \min_{\mathbf{w}} \sum_{i \in P_j} \mathcal{L}(1, \mathbf{w}^\top \mathbf{x}_i) + \sum_{i \in E_j} \mathcal{L}(0, \mathbf{w}^\top \mathbf{x}_i) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2,$$

where λ is the regularization coefficient, $\mathcal{L}(\cdot, \cdot)$ is the point-wise loss function (e.g., squared hinge loss).

Inference: PECOS leverages the tree structure to run beam search algorithm over the tree. In particular, the beam search algorithm chooses a subset of nodes based on the largest scores of the current active nodes, and continues going down from these nodes to arrive at the leaf nodes or HS codes which are relevant for the query. We pick the HS code with the maximum relevance score as the output prediction.

More details about PECOS can be found in Yu et al. [62].

6 ANNOTATION

In this section, we describe the process of collecting ground truth data of product to EIF mappings via manual annotations. Identifying the best match from thousands of EIFs is a challenging and time-consuming task. To reduce burden, we rank the top-5 EIFs using FlamingoZero, and ask annotators if any of these are a match. In case there are <5 EIFs after filtering based on HS codes, we add the top ranked EIFs from the full set until we have a total of 5 options. We also include the option ‘No match’ and ‘Not sure’ to capture both missing EIFs in the database and uncertainty in annotation. We acknowledge that such an annotation system does not measure recall accurately as there could be EIFs in the database which may be an appropriate match, but are not captured in the top ranked EIFs by Flamingo. Nevertheless, it does capture the precision of the algorithm accurately. The recall of the ranked list can be further improved by using a mix of different algorithms such as BM25 [55], universal sentence encoder [8], or with use of late interaction models such as ColBERT [29] that re-rank a candidate list. Our contribution and focus here is on improving the precision of identifying if an EIF is an appropriate match after it has been ranked by a search method. We defer measuring and improving recall of EIF ranking to future work, we expect our results to improve further if recall improves. All our baseline methods, except BM25, rely on the same SBERT model for EIF ranking. Therefore, the results still provide a controlled experiment to compare methods.

In our instructions to annotators, we provide multiple examples of product to EIF mapping, including those with ‘no match’. More than 80% of randomly sampled products do not have an appropriate EIF in the database. To balance this skew, our dataset for annotation includes all the products for which we find an EIF, and randomly sample an equivalent number of products for which there is no match. We use the top-5 ranked EIFs without an HS code filter for these products for a consistent annotation experience.

We use three independent annotations per product and use the majority vote to reduce labeling noise. Our annotators are experienced in labeling tasks. Each annotation took 30 seconds on average. Despite our efforts to reduce annotation burden, our annotators provided feedback that the task was challenging. They had to constantly look up technical terms, such as ‘kenaf’ and ‘mercerizing’: some tasks took as much as 5 minutes. Of the 967 products annotated, 28% had unanimous agreement, 44% had a valid majority vote, and the remaining had split votes with no clear majority. Another 5.6% of the products had ‘Not sure’ as the majority vote. We only consider the 664 products which

Table 3. Performance comparison of HS6 code prediction given product description as input. There are 610K and 152K products in the train and test set respectively.

| Model | Accuracy (%) | Macro F1 | Weighted F1 |
|--------------------|--------------|----------|-------------|
| SBERT (zero-shot) | 11.7 | 9.1 | 11.4 |
| PECOS (supervised) | 81.9 | 44.8 | 81.1 |

have a valid majority, and use the majority voted label as the ground truth for our experiments. The final product dataset has 58% ‘no match’, and 48 unique EIFs with a long-tail distribution.

Krippendorff’s Alpha is a measure of inter-annotator agreement for classification tasks, with 0 and 1 representing perfect disagreement and agreement respectively [30]. The Alpha for our annotations is 0.28, which is similar to values reported for long-tailed classification tasks in the literature [25].

To further validate our dataset, we asked two LCA experts to annotate a subset of 90 products. Picking one of the experts arbitrarily as the ground truth, we measure the precision of both our expert and non-expert annotations. We find that non-experts have a precision of 78.6% on average, and are comparable to expert precision of 76.1%. With majority vote, the non-expert agreement with expert annotations increase to 85.9%. Therefore, we consider our non-expert annotations to be of sufficient quality to be considered as ground truth. We present the full results in Table 5.

7 EVALUATION

We start with results of predicting the HS6 code for a product using the PECOS model, and compare it with zero-shot SBERT model predictions. Next, we analyze the results of predicting EIFs for products for all the baselines and variants of the Flamingo algorithm. We analyze the results from the small 90 product dataset annotated by experts, and then expand the analysis to the larger 664 product dataset. After this analysis, we assess the use of industry codes other than HS codes as the intermediary layer in Flamingo. We then provide an analysis of the prediction errors, and their impact on the carbon footprint values. We end with a micro-benchmark evaluation of Flamingo algorithm on the food ingredient datasets, and compare results with baselines.

7.1 Prediction of HS6 code

We start with evaluation of the HS code classifier M_H on the product to HS code dataset D_H . We split the dataset on a ratio of 4:1 to get train- ($|D_H^{train}| = 610K$) and test- ($|D_H^{test}| = 152K$) sets respectively. We compare the performance of the SBERT zero-shot model M_H^Z as well as the supervised PECOS model M_H^P on D_H^{test} . We pick the prediction with the highest score as the HS6 code, where the score is cosine similarity in case of M_H^Z and confidence score for M_H^P .

We report accuracy, macro F1 score, and weighted F1 score. The F1 scores take into consideration the class-wise accuracy rather than overall accuracy. The macro F1 score gives equal weight to all classes, whereas the weighted F1 score considers the number of products per class. The results are given in Table 3. As we expect, supervision outperforms the zero-shot pre-trained model by a large margin. However, there is a significant distribution shift between products and EIFs. As we show in the next section, the distribution shift impacts the overall performance of the Flamingo algorithm.

7.2 Prediction of EIF

We evaluate EIF predictions by BM25 and SBERT as our baseline algorithms. For BM25, we use the implementation by Trotman et al. [55], and use default hyper-parameters. For both BM25 and

Table 4. Impact of distance threshold on performance of baseline algorithms BM25 and SBERT

| Model | Distance threshold | Precision@1 (%) |
|-------|--------------------|-----------------|
| BM25 | 20 | 29.7 |
| BM25 | 40 | 54.5 |
| BM25 | 60 | 56.9 |
| BM25 | 80 | 58.2 |
| BM25 | 100 | 58.6 |
| BM25 | 120 | 58.1 |
| SBERT | 0.2 | 25.9 |
| SBERT | 0.3 | 29.1 |
| SBERT | 0.4 | 49.5 |
| SBERT | 0.5 | 66.6 |
| SBERT | 0.6 | 61.5 |

Table 5. Evaluation results for Flamingo with a dataset of 90 products. Ground truth is obtained through annotations by an LCA expert.

| Method | Distance threshold | Precision@1 (%) | | | Macro Precision | Weighted Precision |
|-------------------------|--------------------|-----------------|-------------|-------------|-----------------|--------------------|
| | | Overall | No Match | Match | | |
| BM25 | 35 | 65.6 | 77.9 | 27.3 | 15.9 | 65.6 |
| SBERT | 0.5 | 61.1 | 64.7 | 50.0 | 24.4 | 67.9 |
| SBERT + BM25 | 0.5 | 76.7 | 95.6 | 18.2 | 21.8 | 70.0 |
| FlamingoPecos | | | | | | |
| FlamingoPecosHS2 | 0.0 | 58.9 | 66.2 | 36.3 | 21.9 | 74.2 |
| FlamingoPecosHS4 | 0.0 | 75.6 | 89.7 | 31.8 | 23.4 | 76.0 |
| FlamingoPecosHS6 | 0.0 | 72.2 | 91.2 | 13.6 | 15.8 | 68.9 |
| FlamingoPecosHS2 | 0.5 | 76.7 | 89.7 | 31.8 | 28.5 | 75.3 |
| FlamingoPecosHS4 | 0.5 | 81.1 | 97.1 | 31.8 | 29.1 | 73.4 |
| FlamingoPecosHS6 | 0.5 | 77.8 | 98.5 | 13.6 | 18.9 | 68.4 |
| FlamingoZero | | | | | | |
| FlamingoZeroHS2 | 0.0 | 53.3 | 55.9 | 45.5 | 20.4 | 74.9 |
| FlamingoZeroHS4 | 0.0 | 67.8 | 77.9 | 36.4 | 21.7 | 69.2 |
| FlamingoZeroHS6 | 0.0 | 74.4 | 94.1 | 13.6 | 17.1 | 61.6 |
| FlamingoZeroHS2 | 0.5 | 68.9 | 76.5 | 45.5 | 28.8 | 73.9 |
| FlamingoZeroHS4 | 0.5 | 75.6 | 88.2 | 36.4 | 29.7 | 71.0 |
| FlamingoZeroHS6 | 0.5 | 77.8 | 98.5 | 13.6 | 19.9 | 62.2 |
| Human Performance | | | | | | |
| Non-expert (Mean of 3) | – | 78.6 | 82.3 | 68.0 | 46.1 | 85.6 |
| Non-expert (Majority) | – | 85.9 | 88.2 | 79.2 | 60.1 | 88.3 |
| Expert | – | 76.1 | 77.9 | 70.8 | 63.6 | 90.4 |

SBERT, we use a distance threshold that maximizes their overall performance, and include results of threshold sweep in Table 4.

For Flamingo, we include the HS2, HS4, and HS6 based predictions of both FlamingoZero and FlamingoPecos variants. We consider two cosine similarity thresholds for SBERT, 0 (conservative) and 0.5 (aggressive). These choices show the trade-offs in design choices of the Flamingo algorithm, and we avoid tuning these parameters on our dataset.

We consider two datasets, a small dataset of 90 products labeled by an LCA expert as well as annotators, and a larger dataset of 664 products labeled only by the non-expert annotation team. As the annotation is performed on a ranked list of EIFs provided by FlamingoZero, recall cannot be accurately measured. Therefore, we report overall Precision@1, Macro Precision, and Weighted Precision scores. Precision@1, or simply precision, indicates the metric is for the top-ranked candidate, and could be generalized to Precision@K for top-K items. It is a common metric used for text ranking [27]. We also breakdown the overall precision by ‘No match’ and ‘Match’. By design, >50% of the products in the dataset do not have a matching EIF to reflect the importance of predicting ‘No match’ in practice.

Table 5 shows the results on the 90 product dataset. Both BM25 and SBERT perform quite well, giving 65.6% and 61.1% Precision@1 respectively. The choice of threshold determines the trade-off between increasing the precision of ‘Match’ vs ‘No match’. It is intuitive to pick a threshold of 0.5 for SBERT as 0 translates to orthogonal vectors, and 1 to perfectly aligned vectors in embedding space. On the other hand, a good threshold for BM25 changes with the dataset. Flamingo provides an additional method to navigate the same trade-off with HS codes based classification as an intermediate layer. In this dataset, Flamingo helps improve performance by increasing the ‘No match’ precision, and filtering out erroneous EIFs based on their HS code sector.

Flamingo increases the probability of ‘No match’ as we increase the specificity of HS codes from 2 to 6 digits. However, it comes at the cost of decreased ‘Match’ precision. HS4 codes provide a good trade-off between the two extremes, and yield the best overall performance. Addition of cosine similarity threshold on EIF selection further increases the probability of a ‘No match’ prediction. The threshold can be adjusted based on the requirements of downstream applications.

Both FlamingoZero and FlamingoPecos yield similar results, and outperform the baseline algorithms. FlamingoPecos gives a small performance improvement over FlamingoZero. This is in contrast to the large difference in the performance of the HS6 classifiers in Table 3. As the PECOS model M_H^P is trained on the product to HS6 dataset D_H , we hypothesize that FlamingoPecos has low performance on domain specific EIFs such as chemicals and industrial processes. However, as we are using M_H^P to only match EIFs for products, there is still a significant distribution overlap between the datasets D_H and D_P . On the other hand, the SBERT model is trained on a large-scale web corpus, and it can generalize to domain specific text. Use of the zero-shot approach also reduces reliance on external data, and maintenance overhead of training models.

The human performance metrics show there is a significant room for improvement in EIF matching algorithms, especially in selection of an appropriate EIF when it is available in the database. Even when cosine similarity threshold is set to 0, and no EIFs are filtered based on HS codes, the best ‘Match’ precision is 50%, far below the human-level precision of ~70%. This observation points to an opportunity to improve on the SBERT model, perhaps by fine-tuning on sentences used in EIFs.

Table 6 shows the same set of the results for the larger 664-product dataset. In this case, the ground truth is obtained through majority vote among three annotations per product. We include the human-level performance metrics by comparing the annotations of an individual worker against the majority vote. The reported numbers are average values across workers who annotated at least

Table 6. Evaluation results for Flamingo with a dataset of 664 products. Ground truth is obtained by majority vote on three annotations per product.

| Method | Distance threshold | Precision@1 (%) | | | Macro Precision | Weighted Precision |
|--------------------------|--------------------|-----------------|-------------|-------------|-----------------|--------------------|
| | | Overall | No Match | Match | | |
| BM25 | 35 | 53.0 | 87.1 | 5.4 | 7.8 | 46.0 |
| SBERT | 0.5 | 66.6 | 85.8 | 39.7 | 13.9 | 66.8 |
| SBERT + BM25 | 0.5 | 59.0 | 99.0 | 3.2 | 8.1 | 44.5 |
| FlamingoPecos | | | | | | |
| FlamingoPecosHS2 | 0.0 | 62.7 | 68.7 | 54.1 | 13.8 | 80.5 |
| FlamingoPecosHS4 | 0.0 | 57.4 | 89.4 | 12.6 | 12.4 | 69.5 |
| FlamingoPecosHS6 | 0.0 | 57.2 | 92.0 | 8.7 | 7.3 | 67.2 |
| FlamingoPecosHS2 | 0.5 | 68.4 | 92.2 | 35.0 | 14.8 | 72.5 |
| FlamingoPecosHS4 | 0.5 | 61.1 | 97.4 | 10.5 | 11.4 | 49.9 |
| FlamingoPecosHS6 | 0.5 | 59.8 | 97.7 | 6.9 | 6.5 | 47.8 |
| FlamingoZero | | | | | | |
| FlamingoZeroHS2 | 0.0 | 67.2 | 78.3 | 51.6 | 18.0 | 71.3 |
| FlamingoZeroHS4 | 0.0 | 75.0 | 96.3 | 45.1 | 16.4 | 64.0 |
| FlamingoZeroHS6 | 0.0 | 61.4 | 98.7 | 9.4 | 11.0 | 55.2 |
| FlamingoZeroHS2 | 0.5 | 67.8 | 93.8 | 31.4 | 17.3 | 62.9 |
| FlamingoZeroHS4 | 0.5 | 70.2 | 99.2 | 29.6 | 15.7 | 60.7 |
| FlamingoZeroHS6 | 0.5 | 59.5 | 99.7 | 3.2 | 9.4 | 54.2 |
| Human Performance | | | | | | |
| Non-expert (Mean) | – | 76.4 | 76.1 | 75.5 | 42.0 | 86.2 |

100 products. While this is an approximate measure of human performance, it is still instructive to compare these metrics against model predictions.

The trends observed in the small dataset largely hold true for the larger dataset results. However, an exception is that the performance of FlamingoPecos degrades in comparison to both the baseline SBERT algorithm as well as FlamingoZero. We hypothesize that the domain shift is more significant in the larger product dataset. However, FlamingoZero still outperforms the baseline across all metrics, with up to 8.4% improvement in overall precision. Both HS2 and HS4 code based filtering give a good balance between ‘No match’ and ‘Match’ prediction. Similar to the smaller dataset, the largest gap from human performance stems from mis-prediction of EIFs when there is a match in the database. We use the variant FlamingoZeroHS4 with a cosine similarity threshold of 0 as our default algorithm for the rest of the analysis.

7.3 Alternatives to HS code prediction

There are multiple industry classification systems that can be an alternative to HS codes. We look at two choices: International Standard Industrial Classification (ISIC) [36] and Central Product Classification (CPC) [37]. They are pertinent choices as they are already included as part of the EIF metadata in the Ecoinvent database [57]. However, we do not have the ISIC or CPC codes of the products in our dataset, so we use the SBERT model to predict the best ISIC and CPC codes for each product using text descriptions. The United Nations also publishes correspondence tables

Table 7. Performance of EIF matching using different industry sector codes

| Industry Code | Distance threshold | Precision@1 (%) |
|---------------|--------------------|-----------------|
| HS4 | 0.0 | 75.0 |
| HS4 | 0.5 | 70.2 |
| ISIC | 0.0 | 63.4 |
| ISIC | 0.5 | 68.7 |
| CPC | 0.0 | 52.7 |
| CPC | 0.5 | 59.9 |
| CPC to HS2 | 0.0 | 50.6 |
| CPC to HS2 | 0.5 | 57.8 |






| Product Description | Product HS code | Best EIF by Semantic Matching | EIF HS code | Human Label | Type of error |
|---|---------------------------------------|--------------------------------|---------------------------------------|---------------------------|-----------------------------|
|  Lithium coin cell | Cells and batteries | Graphite, battery grade | Graphite | Battery cell, Li-ion | EIF mismatch |
|  High-yield toner cartridge | Printing machinery | Toner module; black/white | Ink; for printing | Toner module; color | HS code mismatch |
|  Standard duct tape | Plastics | Sealing tape; aluminum/PE | Aluminum | Sealing tape; aluminum/PE | Human error |
|  USB cable | Insulated electric conductors | Network cable, category 5 | Connectors for optical fibers | Cable, unspecified | No fitting HS code |
|  Heavy duty filter cartridge | Machinery; for filtering or purifying | Air filter, decentralized unit | Machinery; for filtering or purifying | No match | No EIF for matching HS code |

Fig. 7. Examples of errors by Flamingo. Most errors can be categorized into the five types shown.

between CPC and HS codes, we include this as an additional option to identify HS codes for EIFs instead of predicting with the SBERT model. Table 7 shows the performance metrics on the 664 product dataset.

EIF matching based on HS codes outperform the rest of the options across all metrics. Upon manual inspection, we find that HS6 codes are more granular with a deeper hierarchy compared to ISIC and CPC. The assignment of ISIC and CPC codes in the EIF database is also subjective, e.g., some mappings correspond to 3 digit CPC codes while others correspond to 5 digits. There are also errors in mapping between CPC and HS codes in the correspondence tables, and directly predicting the HS codes reduces the errors compared to mapping correspondence data across two sources.

7.4 Analysis of errors

We randomly sampled 50 data points incorrectly predicted by the FlamingoZeroHS4 from Table 6, and manually analyzed them to understand the reasons behind the erroneous outputs. Figure 7 shows a few examples of the errors, categorized into five types. A majority of the errors (40%) corresponded to electronic cables such as those for charging or connecting components (row 4

in the figure). There is an EIF in Ecoinvent, called ‘cable, unspecified’, which captures all of these different types of cables. We find that SBERT is poor at matching text such as these, where it needs to understand the set relationship across closely related EIFs or HS codes. Some HS codes contain descriptions of similar nature, e.g., ‘not elsewhere classified’, and are a common source of errors in matching. Methods which capture such set relationships or exploit the hierarchy in a different manner could overcome this issue.

In some cases (22%, row 2 in figure), the error is due to mismatch between HS codes even when the EIF predicted is the same as the selected ground truth. For example, a toner module is mapped to the HS code ‘ink for printing’ by the SBERT model, whereas the product maps it to a related HS code ‘printing machinery’. In this case, the two related HS codes are not close to each other in the hierarchy, so using a 2-digit or 4-digit version of the code did not remove the error. A related source of error (8%, row 5 in figure) is when EIF predicted is not a match even when the HS codes match. In this case, the HS code category is too broad and does not differentiate between unrelated EIFs. Such errors point to the limitation of using HS codes as an imperfect classification system. A data-driven knowledge base can potentially address such issues by capturing relationships across concepts in a multi-dimensional manner.

About 20% of the errors were due to semantic matching errors (row 1 of figure), and point to opportunities for improving on SBERT models. There were a few errors due to human mislabeling as well (10%, row 3 of figure), which could be reduced by improving the annotation procedure. Finally, we found an intriguing but rare error not shown in Figure 7. In one case, the product was a package of keyboard and mouse. The humans annotated the ground truth EIF as keyboard, whereas Flamingo predicts the EIF as mouse. Such errors can be addressed if we break down composite products to their individual items.

7.5 Impact of prediction error on carbon footprint

We analyze the impact of an EIF misprediction on downstream use cases using the corresponding greenhouse gas emission values in units of kilograms of carbon dioxide equivalent (kgCO_2e). We used the ‘reference product name’ metadata in the Ecoinvent dataset to represent an EIF, but there could be multiple entries for a single reference product based on variations such as region of manufacture or system boundary. For this analysis, we compute the average of these individual values following similar practice from prior works [11, 33]. We use the metrics Pearson’s correlation (Pearson R), mean absolute percentage error (MAPE), and median absolute deviation (MAD) to characterize the errors.

We use the predictions by FlamingoZeroHS4 variant of the algorithm for this evaluation. Our analysis is limited by the known EIFs in the dataset. Of the 664 products in our dataset, we only have ground truth EIF for 277 products as the rest do not have a match. Of these 277 products, 130 are predicted to have a match by FlamingoZeroHS4. 125 of the 130 products are predicted correctly, and yield low error – Pearson R: 0.95, MAPE: 18%, MAD: 0 kgCO_2e . This statistic indicates that misprediction of ‘no match’ are the major source of errors.

To further understand the impact of errors due to semantic matching, we analyze the error after removing the HS code based filter. Predictions for 172 of 277 products are correct, and the corresponding error metrics are – Pearson R: 0.81, MAPE: 144%, MAD: 3.9 kgCO_2e . Figure 8 compares the carbon emission values of the ground truth EIF to that of the predicted ones. We use a log/log plot to cover the wide range of values, trends look similar in linear scale. Overall, there is a high correlation in predicted and ground truth values, the errors are small in absolute terms but larger in relative terms. There are a few anomalous predictions that cause an error in multiple orders of magnitude, four points have >1000% error and an additional 5 points have >100% error.

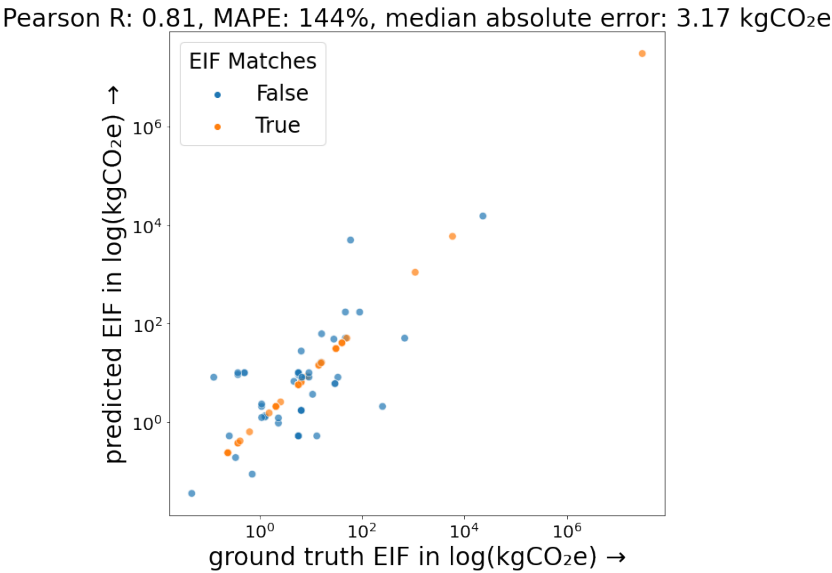


Fig. 8. Impact of EIF misprediction error on downstream carbon footprint estimate. In most cases, the error in prediction does not lead to a large error in carbon emissions, however, there are a few outliers that cause large deviations.

Table 8. Evaluation results for Flamingo with a dataset of 264 food ingredients. Ground truth is obtained through annotation by a non-expert worker.

| Method | Distance threshold | Precision@1 (%) | | | Macro Precision | Weighted Precision |
|-----------------|--------------------|-----------------|----------|-------------|-----------------|--------------------|
| | | Overall | No Match | Match | | |
| BM25 | 0.1 | 23.1 | 19.8 | 30.0 | 13.4 | 56.9 |
| SBERT | 0.5 | 34.8 | 14.1 | 77.0 | 26.1 | 86.2 |
| FlamingoZeroHS4 | 0.0 | 70.0 | 75.7 | 57.4 | 41.0 | 76.1 |

Some examples of errors include predicting a microwave as HVAC unit (8118%), incorrect battery chemistry (324%), and incorrect paper type (112%).

7.6 Generalization to food ingredients

LCA experts often match EIFs to materials, manufacturing processes, and not just products. We evaluate if Flamingo generalizes to use cases beyond our product dataset using a separate dataset of food ingredients. Unlike the product descriptions, the ingredients consist of only a few words, e.g. ‘organic diced tomatoes’, ‘purified water’, ‘organic licorice’. Our dataset consists of 272 ingredients, and we follow the same steps as Section 6 to annotate the ground truth. Due to the labor-intensive nature of annotations, we only use one worker for this micro-benchmark and do not have a consensus based label. Unlike generic products, basic materials such as food ingredients have a higher probability of finding a match in the EIF dataset. From the annotations, we find 87 ingredients that match to EIFs and 177 that do not have a match.

Table 8 summarizes the results. We report the results of the FlamingoZeroHS4 algorithm, and compare it with SBERT and BM25 baselines. In this case, the HS code for the ingredient is provided

as part of the dataset, and we rely on Flamingo to predict the HS codes for both EIFs and ingredients. The results follow the same trends as those from product dataset experiments, although the improvement compared to baselines is even larger in this case (35.2%). As an example, the best semantic match for ‘pineapple juice’ is ‘pineapple’. The HS codes for these two strings are different, and Flamingo correctly predicts ‘No match’.

From an experience point of view, LCA experts find Flamingo to be useful and report significant time savings. A skilled practitioner used Flamingo to complete LCA of 15K food products in just 15 minutes, a task that would have previously taken 40 hours of manually mapping EIFs.

8 CONCLUSION AND FUTURE WORK

Measurement of greenhouse gas (GHG) emissions of a consumer products is a critical step toward understanding mitigation decisions. Life cycle assessment (LCA) provides a systematic approach to measurement of GHG emissions, but is challenging to scale due to manual steps. Identifying an appropriate environment impact factor (EIF) is a critical aspect of LCA, and current practice requires significant time-investment from LCA experts. We have presented an algorithm that automates EIF matching for a given query text. Our algorithm, Flamingo, requires no training data and exploits industry codes to predict whether an EIF exists in the database as well as identifies the best matching EIF when it exists. Evaluation on a dataset of 664 products shows that Flamingo can achieve 75% precision, and predicts when no EIF exists with 96% precision. While we have focused on GHG emissions throughout the paper, the EIF databases include impact estimates for additional categories such as hazardous wastes, fresh water use, and air pollution. Our algorithms can be easily extended to these impacts.

There are several directions to explore which can improve on our methods. We can extend text-based matching to image-based matching using CLIP [39], which generates an embedding with an image-to-text correspondence. The key to further improvement in EIF matching performance is to improve the prediction of HS codes from EIF metadata. While the supervised approach did not lead to significant improvement in performance, future work can consider zero-shot contrastive learning approaches such as MACLR [59] that can exploit EIF text descriptions to learn correspondence with HS codes.

While automation of LCA has been studied for several decades [2], it is now starting to be adopted on a large scale to comply with regulations, meet customer demand, and combat climate change. Flamingo provides a promising start on a critical aspect of LCA, but a number of challenges remain in scaling to millions of consumer products. As seen from our results, EIFs of most consumer products in the market are not available in the database, and more EIFs are added manually from publicly available documents. Automation of EIF extraction from documents is a promising avenue of future work.

ADDITIONAL REMARKS

Aravind Srinivasan’s contribution to this publication was not part of his University of Maryland duties or responsibilities.

ACKNOWLEDGMENT.

We thank the referees for their several helpful comments and suggestions.

REFERENCES

- [1] International Energy Agency. 2021. Global Energy Review 2021: Assessing the effects of economic recoveries on global energy demand and CO2 emissions in 2021. <https://www.iea.org/reports/global-energy-review-2021>. (2021).

- [2] Mikaela Algren, Wendy Fisher, and Amy E Landis. 2021. Machine learning in life cycle assessment. In *Data Science Applied to Sustainability Analysis*. Elsevier, 167–190.
- [3] Bharathan Balaji, Venkata Sai Gargeya Vunnava, Geoffrey Guest, and Jared Kramer. 2023. CaML: Carbon Footprinting of Household Products with Zero-Shot Semantic Text Similarity. In *Proceedings of the ACM Web Conference 2023*. 4004–4014.
- [4] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code. <http://manikvarma.org/downloads/XC/XMLRepository.html>
- [5] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems* 28 (2015).
- [6] Robert S Boyer and J Strother Moore. 1977. A fast string searching algorithm. *Commun. ACM* 20, 10 (1977), 762–772.
- [7] Vannevar Bush et al. 1945. As we may think. *The Atlantic Monthly* 176, 1 (1945), 101–108.
- [8] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*. 169–174.
- [9] Peggy Chaplin. 1987. An Introduction to the Harmonized System. *NCJ Int’l L. & Com. Reg.* 12 (1987), 417.
- [10] Xi Chen, Stefano Bromuri, and Marko Van Eekelen. 2021. Neural machine translation for harmonized system codes prediction. In *2021 6th International Conference on Machine Learning Technologies*. 158–163.
- [11] Michael Clark, Marco Springmann, Mike Rayner, Peter Scarborough, Jason Hill, David Tilman, Jennie I Macdiarmid, Jessica Fanzo, Lauren Bandy, and Richard A Harrington. 2022. Estimating the environmental impacts of 57,000 food products. *Proceedings of the National Academy of Sciences* 119, 33 (2022), e2120584119.
- [12] Vincent Colomb, Samy Ait Amar, Claudine Basset Mens, Armelle Gac, Gérard Gaillard, Peter Koch, Jerome Mousset, Thibault Salou, Aurélie Tailleux, and Hays MG van der Werf. 2015. AGRIBALYSE®, the French LCI Database for agricultural products: high quality data for producers and environmental labelling. *OCL* 22, 1 (2015), D104.
- [13] Akshay Raj Dhamija, Manuel Günther, and Terrance Boulton. 2018. Reducing network agnostophobia. *Advances in Neural Information Processing Systems* 31 (2018).
- [14] Liya Ding, ZhenZhen Fan, and DongLiang Chen. 2015. Auto-categorization of HS code using background net approach. *Procedia Computer Science* 60 (2015), 1462–1471.
- [15] Shaohua Du, Zhihao Wu, Huaiyu Wan, and YouFang Lin. 2021. HScodeNet: Combining Hierarchical Sequential and Global Spatial Information of Text for Commodity HS Code Classification. In *Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part II*. Springer, 676–689.
- [16] International Organization for Standardization. 2006. *Environmental management: life cycle assessment; Principles and Framework*. ISO.
- [17] Tao Gao, Qing Liu, and Jianping Wang. 2014. A comparative study of carbon footprint and assessment standards. *International Journal of Low-Carbon Technologies* 9, 3 (2014), 237–243.
- [18] Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O’Reilly Media, Inc."
- [19] Jeroen B Guinée, Reinout Heijungs, Gjalt Huppes, Alessandra Zamagni, Paolo Masoni, Roberto Buonamici, Tomas Ekvall, and Tomas Rydberg. 2011. Life cycle assessment: Past, present, and future. *Environmental Science and Technology* 45, 1 (2011), 90–96.
- [20] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *Advances in neural information processing systems* 27 (2014).
- [21] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6529–6537.
- [22] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.
- [23] Robert G Hunt. 1974. *Resource and environmental profile analysis of nine beverage container alternatives*. Vol. 91. Environmental Protection Agency.
- [24] Wesley Ingwersen, Maria Gausman, Annie Weisbrod, Debalina Sengupta, Seung-Jin Lee, Jane Bare, Ed Zanolli, Gurbaksh S Bhandar, and Manuel Ceja. 2016. Detailed life cycle assessment of Bounty® paper towel operations in the United States. *Journal of cleaner production* 131 (2016), 509–522.
- [25] Hamid Jalalzai, Pierre Colombo, Chloé Clavel, Eric Gaussier, Giovanna Varni, Emmanuel Vignon, and Anne Sabourin. 2020. Heavy-tailed representations, text polarity classification & data augmentation. *Advances in Neural Information Processing Systems* 33 (2020), 4295–4307.

- [26] Lucas Joppa, Amy Luers, Elizabeth Willmott, S Julio Friedmann, Steven P Hamburg, and Rafael Broze. 2021. Microsoft's million-tonne CO₂-removal purchase—lessons for net zero.
- [27] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 427–431.
- [28] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
- [29] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.
- [30] Klaus Krippendorff. [n. d.]. Computing Krippendorff's Alpha-Reliability. *Computing* 1 ([n. d.]), 25–2011.
- [31] Eunji Lee, Sundong Kim, Sihyun Kim, Sungwon Park, Meeyoung Cha, Soyeon Jung, Suyoung Yang, Yeonsoo Choi, Sungdae Ji, Minsoo Song, et al. 2021. Classification of goods using text descriptions with sentences retrieval. *arXiv preprint arXiv:2111.01663* (2021).
- [32] Christoph J Meinenken, Daniel Chen, Ricardo A Esparza, Venkat Iyer, Sally P Paradis, Aruna Prasad, and Erika Whillas. 2022. The Carbon Catalogue, carbon footprints of 866 commercial products from 8 industry sectors and 5 continents. *Scientific Data* 9, 1 (2022), 87.
- [33] Christoph J Meinenken, Scott M Kaufman, Siddharth Ramesh, and Klaus S Lackner. 2012. Fast carbon footprinting for large product portfolios. *Journal of Industrial Ecology* 16, 5 (2012), 669–679.
- [34] Bhaskar Mitra, Nick Craswell, et al. 2018. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval* 13, 1 (2018), 1–126.
- [35] Andrew Moore. 2001. K-means and Hierarchical Clustering.
- [36] United Nations. 1969. *International standard industrial classification of all economic activities*. UN.
- [37] Peter Pariag. 2009. Classification of services. In *Regional Symposium on Services*. 15–17.
- [38] Hans-Otto Pörtner, Debra C Roberts, H Adams, C Adler, P Aldunce, E Ali, R Ara Begum, R Betts, R Bezner Kerr, R Biesbroek, et al. 2022. Climate change 2022: Impacts, adaptation and vulnerability. *IPCC Sixth Assessment Report* (2022), 37–118.
- [39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
- [40] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
- [41] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. Citeseer, 29–48.
- [42] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.
- [43] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
- [44] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.
- [45] Davide Rovelli, Carlo Brondi, Michele Andreotti, Elisabetta Abbate, Maurizio Zanforlin, and Andrea Ballarino. 2022. A Modular Tool to Support Data Management for LCA in Industry: Methodology, Application and Potentialities. *Sustainability* 14, 7 (2022), 3746.
- [46] Amit Singhal et al. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.
- [47] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems* 33 (2020), 16857–16867.
- [48] Ines Sousa and David Wallace. 2006. Product classification to support approximate life-cycle assessment of design concepts. *Technological Forecasting and Social Change* 73, 3 (2006), 228–249.
- [49] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.
- [50] Sphera. 2023. Life Cycle Assessment Product Sustainability (GaBi) Software. <https://sphera.com/life-cycle-assessment-lca-software/>. (2023).
- [51] GHG Protocol Standard. 2011. The greenhouse gas protocol.
- [52] Zequn Sun, Muhao Chen, and Wei Hu. 2021. Knowing the No-match: Entity Alignment with Dangling Cases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3582–3593.

- [53] Tomohiro Tasaki, Koichi Shobatake, Kenichi Nakajima, and Carl Dalhammar. 2017. International survey of the costs of assessment for environmental product declarations. *Procedia CIRP* 61 (2017), 727–731.
- [54] Ken Thompson. 1968. Programming techniques: Regular expression search algorithm. *Commun. ACM* 11, 6 (1968), 419–422.
- [55] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*. 58–65.
- [56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [57] Gregor Wernet, Christian Bauer, Bernhard Steubing, Jürgen Reinhard, Emilia Moreno-Ruiz, and Bo Weidema. 2016. The ecoinvent database version 3 (part I): overview and methodology. *The International Journal of Life Cycle Assessment* 21, 9 (2016), 1218–1230.
- [58] Hans-Michael Wolffgang and Christopher Dallimore. 2011. The World Customs Organization and its Role in the System of World Trade: An Overview. *European Yearbook of International Economic Law* 2012 (2011), 613–633.
- [59] Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. 2022. Extreme Zero-Shot Learning for Extreme Text Classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5455–5468.
- [60] Yi Yang, Wesley W Ingwersen, Troy R Hawkins, Michael Srocka, and David E Meyer. 2017. USEEIO: A new and transparent United States environmentally-extended input-output model. *Journal of cleaner production* 158 (2017), 308–318.
- [61] Hsiang-Fu Yu, Jiong Zhang, Wei-Cheng Chang, Jyun-Yu Jiang, Wei Li, and Cho-Jui Hsieh. 2022. Pecos: Prediction for enormous and correlated output spaces. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4848–4849.
- [62] Hsiang-Fu Yu, Kai Zhong, and Inderjit S. Dhillon. 2020. PECOS: Prediction for Enormous and Correlated Output Spaces. arXiv:2010.05878 [cs.LG]
- [63] Weixin Zeng, Xiang Zhao, Jiuyang Tang, Xinyi Li, Minnan Luo, and Qinghua Zheng. 2021. Towards entity alignment in the open world: An unsupervised approach. In *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I* 26. Springer, 272–289.
- [64] Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-Task Label Embedding for Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4545–4553.

accepted 13 July 2023