Using Language Model for Emission Factor Mapping

Stanford CS224N Custom Project

Gerald Kang
Department of Computer Science
Stanford University
gkang03@stanford.edu

Esteban Barrero-Hernandez
Department of Computer Science
Stanford University
ejbh24@stanford.edu

Amar Gadkari
Department of Computer Science
Stanford University
amarga@stanford.edu

Abstract

As climate change continues to pose a significant threat to global stability, the precision in tracking and reporting carbon emissions has never been more crucial. Our project aims to enhance the accuracy and efficiency of carbon emission tracking for businesses using a language model. The goal is to minimize human labor in data mapping processes and improve the reliability of emission data. In order to tackle this problem, we developed a fine-tuned BERT model on synthetic business activity data which is able to properly map divergent user carbon emission tags to their golden standard. This synthetic fine-tuning data was generated through a conjunction of a Python script that randomly samples column examples returned by ChatGPT. This simulates business activity data which we further introduced deterministic error to replicate human mistakes. With our initial Random Forest Classifier, we were able to achieve 90% accuracy with initial set of noise injected in the input data. When we improved the noise injection more controlled the Random Forest Classifier the accuracy was 99.64% while our BERT fine-tuned model boosted this to 99.9% and above.

1 Key Information to include

- Mentor: Caleb Ziems suggested use of bert, and noise generation approach, and helped us focus on measurements.
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Group Contributions: Amar came up with the original project idea, found research related work, created the initial dataset, performed the initial experiments with the RandomForest-Classifier, and created the movie error dataset. Gerry significantly improved our accuracy by integrating the use of a pre-trained BERT model, fine-tuning the model, and testing it on our data. Esteban created the twitter error dataset and introduced a sentence embedding model to score the quality of datasets. All members were significantly involved in progressing the project by researching for new ideas and preparing the final report.

2 Introduction

Carbon reporting remained an after thought for large corporations until fairly recently. Given world-wide focus on climate change the quality of carbon reporting is becoming important from regulations perspective, but even more critical from brand recognition perspective. Carbon reporting expects

to track emissions for products from cradle to grave. This data exists in large corporations but in disparate systems such as enterprise relationship management systems, travel management systems, various other back-office, IT, and HR systems. Today, that business activity data is tagged and classified manually to identify correct emission factor as prescribed by the United States Environmental Protection Agency (EPA) and other equivalent government agencies. This paper focuses on improving the quality and human labor in mapping the business activity data from various IT systems to the correct emission data. We are excited by, recent advancements in large language model specially in text classification area. This paper discusses applying recent advancements in language model to business activity mapping to EPA's emission factor data. Additionally, we share the results for the mapping activity. We specifically focus on understanding error rate in the business activity data large language models can tolerate.

3 Related Work

In recent research conducted at IBM (1), advances have been made in utilizing large language models for estimating supply chain emissions. This work is crucial in simplifying the complex task of carbon emission reporting, particularly for Scope 3 emissions, which can often account for the majority of a company's environmental impact. By fine-tuning BERT models, researchers aim to categorize business activities more accurately to Environmental Input-Output (EEIO) categories, thereby refining the emission factor computation process for these activities.

The IBM study not only enhances the method of data collection for emissions reporting but also compares the efficiency of various models and featurization techniques. Their findings suggest that, despite the challenges of diverse product descriptions across companies, pretrained language models like BERT, Roberta, and ClimateBert significantly outperform traditional machine learning algorithms in emissions classification. This breakthrough presents a promising direction for future research in environmental sustainability, offering a benchmark for models and methodologies that efficiently process and categorize emission-related data.

This body of work, however, acknowledges certain limitations. It points out that while identifying the most effective language model is essential, the overall accuracy of EEIO category mapping can be further improved by incorporating additional transactional parameters. These parameters might include vendor names or categories which can enhance the precision of mapping, especially when product descriptions are ambiguous. Such insights are invaluable as they guide subsequent research to not only focus on model performance but also on the richness of the data used, ensuring more accurate and meaningful environmental reporting.

4 Approach

To approach this classification problem, we utilized two different models: Random Forest Classifier and BERT. While Random Forest Classifier utilizes a tree architecture, BERT employs encoders. Due to their difference in structure, we exercised a fine-tuning approach for BERT whereas a hyperparamter optimization training method for the Random Forest Classifier. These are further explained below:

4.1 Random Forest Classifier

The Random Forest Classifier operates on the principle of ensemble learning; combining the predictions from multiple machine learning algorithms to consolidate them into a singular, more accurate prediction. The ensemble method builds numerous decision trees and merges their outputs leading to an improvement in overall predictive accuracy. The key term "Forest" in Random Forest Classifier highlights this architecture, where this model builds an ensemble of decision trees trained upon the bootstrap aggregating method (training multiple models on different subsets of data, then averaging their results). Because each ensemble is built from a random sample of training, this means that the "best nodes" within that "forest" is only the best for that subset of data. This leads to the Random Forest Classifier, which inherently holds diversity, having innate robustness.

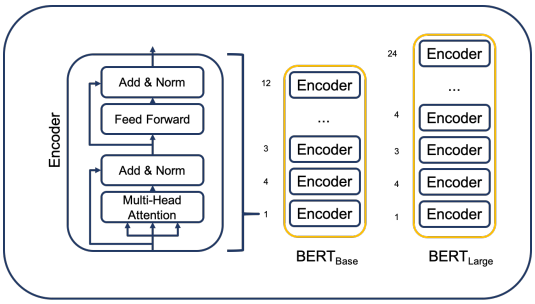

Figure 1: Bert Architecture

The preprocessing and training of the data consisted of converting textual information to its TF-IDF representation. TF-IDF, short for Term Frequency-Inverse Document Frequency, is a numerical statistic reflecting a word's importance based on its frequency in the document and the corpus. This relationship allows us to lessen the effect of commonly used, but trivial words. These transformed inputs are then used for the training of our model (from a newly initialized stage). Furthermore, the Random Forest Classifier holds a hyperparameter tuning process using RandomizedSearchCV which is initiated during training, allowing it to optimize parameters such as the number of estimators, tree depth, and minimum samples for splits/leaves for the best classification accuracy.

4.2 BERT

BERT, short for Bidirectional Encoder Representations for Transformers, is a model relying on an attention mechanism rather than sequence analysis. This allows the model to weigh the influence of different words within a sentence, regardless of its index.

The key to BERT is its multi-layer bidirectional Transformer encoders. A Transformer encoder reads an entire sequence in a certain direction (left to right or right to left) allowing the model to learn a word's context based on its surroundings. Because BERT is bidirectional, meaning a word's context is learned both left to right and right to left, the model's understanding of a word's meaning is more complete and thorough. The attention mechanism within transformers assigns a weight to each word in a sentence based on its contribution to the surrounding context of the predicted word.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Q, K,and V respectively correlate to the query, key, and value matrix while d_k refers to the dimensionality of the key matrix. Mathematically, the attention mechanism computes the dot product of the query with all of the keys to determine the relevance of other words to the current word (i.e. matrix Q). This relevance is then scaled down by dividing by d_k. These probabilities are then distributed along a scale from 0 to 1 through the softmax equation, resulting in an output representing the weighted representation of the input based on its context.

The attention layers within the BERT encoders are followed by a feed-forward network, applying additional transformations to the output matrix. As these transformations are passed down through each encoder, the final layer outputs a cumulative contextual representation of each input token. This output matrix can then be built upon by adding a final layer trained on task-specific data. This process is known as fine-tuning.

For our approach to tackle proper mapping of user carbon emission tags to their golden standard, we fine-tuned a smaller version of BERT ('bert-base-uncased') on our synthetically created, error-induced training set so that the final encoder layer can identify subtler mistakes within business

activity data. An example of a nuance that BERT with pretraining would catch if particular vendor handles packaging and business activity is marked incorrectly as a raw material instead of packaging material, it would correctly classify as a packaging material. The inherently "messy" enterprise data can be systematically self-corrected using power of the language model and specifically pre-training as shown by the results.

5 Experiments

5.1 Data

We generate synthetic business activity data using the python script. The script generates comma separated values (csv) file containing columns described below in detail. The description of the activity, vendor names, comments though fictional but generated with help of GPT 4 to make it close to real world. There was considerable randomness added to avoid overfitting during the training phase. To get the synthetic data closer to real world, we have put considerable consideration in adding the noise. We took words from a movie title database available on kragle and removed punctuation and numerals from the titles. We also discarded all words that are less than 4 character long. This gave us very realistic noise that then we added to the data by replacing only a certain set of words from the description, vendor, and comment columns.

5.2 Dataset Description

Our datasets are supposed to reflect realistic business activities, each associated with specific North American Industry Classification System (NAICS) titles. The enterprise data is expected to be messy. These activities are categorized across several sectors, such as "Support Activities for Metal Mining," "Sewage Treatment Facilities," "New Single-Family Housing Construction," and more, encompassing a wide range of industries. For each activity, the dataset details the vendors involved, costs in USD, and comments highlighting the nature of the activity, including environmental and sustainability considerations. The business activity data includes the following key columns:

Business Activity Description: Describes the nature of the business activity.
Business Activity Vendor: Names the vendors involved in the activity.
Business Activity Cost USD: Lists the cost associated with the activity in U.S. dollars.
Business Activity Comment: Provides additional context or comments about the activity.
2017 NAICS Title: Specifies the industry category to which the activity belongs, according to the 2017 NAICS titles.

5.3 Dataset Creation

We first defined the column names (csv columns) that form the dataset's structure, capturing essential aspects such as business activity descriptions, vendors, associated costs, comments, and the corresponding NAICS titles. We have used a data model (Business Activities) to represent business activities across various industries, detailing the role of activity pertaining to product, vendors involved, and descriptive comments. To dynamically generate individual records reflecting real-world scenarios, we created a generate_record function, which samples business activities and pairs them with randomly generated costs. The randomly generated cost simulate financial variability. We also created a generate_dataset function that creates and aggregates a large number of these records into an extensive dataset (csv file). To support robust model development and testing, we partitioned this dataset into training and test subsets. We did this by selecting a predetermined fraction, 0.83% of the data for training purposes, with the rest of the data allocated for testing. The rows are randomly partitioned to the training and test dataset csv files. We created 13744 examples for the training data 2756 examples for the test data.

5.3.1 Successfully Adding Synthetic Noise to Datasets

We wanted to effectively simulate the presence of errors within the datasets, offering a valuable resource for testing imperfect data which is realistic and expected in the real world. To infuse our datasets with errors, simulating the imperfections commonly found in real-world data, we randomly introduced a range of errors into the text fields of a dataset, from minor typographical errors to

significant inaccuracies, reflecting the nuances of data quality challenges. First, we developed two distinct functions, introduce_minor_errors and introduce_major_errors, to embed synthetic errors within text data. While the former mimics common typing mistakes like substitution, omission, or swapping of characters, the latter drastically alters the text, either by scrambling it or appending irrelevant content, to represent major inaccuracies. We then implemented the apply_random_error function, which, based on a predetermined probability, decides whether to inject an error into a given piece of text. This introduces variability, ensuring that not all records are uniformly affected, thereby mimicking real-world data inconsistency. We also created a apply_errors_with_limit function to randomly apply errors to a limited number of specified text fields within a dataset row. This method ensures a controlled dispersion of errors, preventing the dataset from becoming overwhelmingly inaccurate while still reflecting realistic data quality issues. Finally, we applied to these functions to both our training and test datasets, outputting large datasets with errors to test our model. Our model performs really well on these datasets. It is possible that the randomness we use with Python may be too deterministic which is unlike human errors. This led us to explore other options.

5.3.2 Failing to add Human-Like Noise to Datasets

We planned to use another model to inject human-like errors into our datasets. First, we used a pre-trained LLM, fine-tuned on a Twitter Corpus. We found around 10000 tweets across a couple datasets online, parsed them, and aggregated them into a text file. Then we fine-tuned a pre-trained GPT2 model, which we imported, on this huge file of collected tweets. We thought that the tweets would accurately capture human-like mistakes in language. We looped through each row and relevant column in the datasets, and we provided a prompt with instructions to inject errors into the content along with the content to the model for text generation. Unfortunately, it always failed to regenerate the content with errors. Although the generated text was very Twitter-like, so the fine-tuning work, regardless of how carefully-crafted the prompt and experimenting with other variables, it would only generate new text after the original content and not within. We also tried with a version of of GPT2 that was not fined-tuned and a more powerful pre-trained GPT Neo 2.7 model, but they still failed to understand the prompt. Any more powerful model was off-limits to due price.

This failure to create a strong dataset was reaffirmed when we decided to use another model that is small and could classify our dataset into a score that reflects how realistic these errors are to rate the quality of our datasets with errors and measure how human-like these errors. We used a sentence embedding model which we could feed our dataset without errors and our dataset with errors to This model can output a similarity score for each cell and an average for all scores. A high similarity score would reflect that they datasets are very similar, which reflect that dataset with errors is realistic because human-like errors are usually very similar to the correct language. If we compared the original data with the aesthetic error data and the original data with the error data with more realistic noise, getting a higher score on the latter would reaffirm that we were able to inject realistic noise into our dataset. Unfortunately, this test dataset with noise from Twitter got an average similarity score of 0.701. Our original synthetic test dataset got an average similarity score of around a 1. Thus, this new dataset is less similar to the original dataset than the previous synthetic dataset, proving that it was not a good representation of human-like errors and we had to find a new approach.

5.3.3 Using Movie Title Dataset to Integrate Quality Noise

Initially for the first iteration we added words generated using random characters. That approach only generated 90% success rate. This approach had either very high test and training accuracy or failing poorly. Language model was able to tease apart meaningless errors quickly. As a result, we decided to inject noise from more realistic data. We retrieved movie titles from the movie database available online. By removing obvious noise from the movie names by simply removing numerals, punctuation's, and less than 4 character words, we could extract 33,466 words that we use to inject the noise. For testing purposes we do the controlled noise injection in the business activity data. The error_rate parameter controls percentage of rows in the training and test dataset would have error. Additionally the same parameter controls in a given row percentage of words replaced by noise words. This gives fine control over the noise in the business activity data to help us understand how robust the classifier models are. This approach fared well because RandomForestClassifier can tolerate errors up-to 70% beyond that error rate, the classification accuracy reduces drastically.

5.4 Evaluation method

To test our models, we evaluated their performance based on how accurately the models can predict the correct EPA emission category for each business activity. Specifically, the percentages we provide are the number of data points the model predicts correctly over the total number of data points. Furthermore, as mentioned above, we decided to use a sentence embedding model based on the MiniLM architecture with 6 layers to score the quality of our datasets with errors.

5.5 Experimental details

5.5.1 Random Forest Classifier

The best parameters returned by RandomizedSearchCV was:

- `n_estimators`: 1200, `min_samples_split`: 15, `min_samples_leaf`: 2,
- `max_features`: log2, `max_depth`: 60

5.5.2 BERT

The parameters that returned the best results were:

- `learning_rate`: $1e^{-8}$, `weight_decay`: $2e^{-5}$, `batch_size`: 64,
- `max_len`: 128, `epochs`: 50

5.6 Results

5.6.1 Random Forest Classifier

The RandomForestClassifier achieved 90% accuracy on the synthetic error dataset. After improving our method for injecting errors by using move titles, we tested the RandomForestClassifier on several error rates. We tested on both test data and on the training data that we used to train BERT. For the most part, outcomes were about the same when testing on these 2 datasets. As you can see below, for anything below a 40% error rate, RandomForestClassifier achieved about 100% accuracy. It dips below 99% accuracy at around 66% error rate and below 99% at around 68% error rate. Once the 70% error threshold is met, the accuracy starts to decline faster until its around 80% accurate at about 95% error rate, which is good. After this, the accuracy plummets to 0% as the error rate approaches 100%, which makes sense because this is when every word in the original dataset is replaced with the noise.

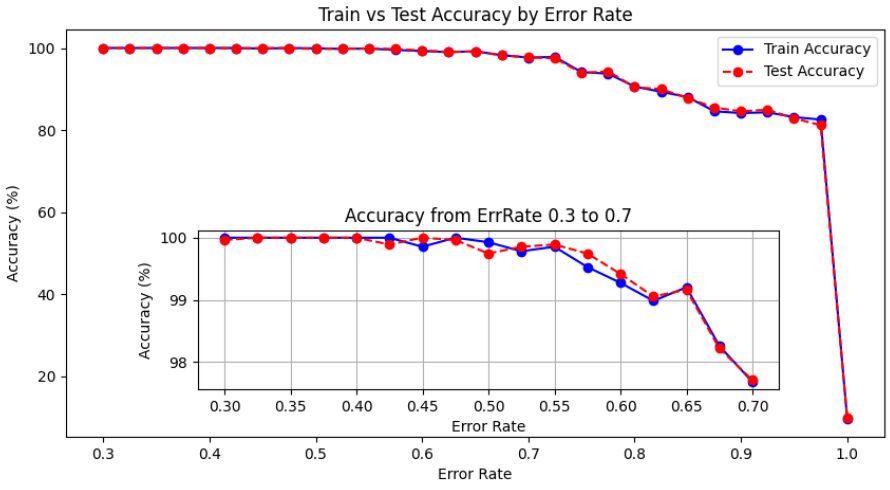

Figure 2: Accuracy on RandomForestClassifier for Error Dataset with Movie Key Words

5.6.2 BERT

When training and testing BERT on the deterministic error data without movies data, BERT exhibits an extraordinary 99.89% accuracy. Furthermore, we graphed the training loss for each epoch below, and it decreased to minimum of around 0.00001. The training time was 54 minutes. However, due to the obvious synthetic nature of our data, we knew this did not capture the model's full ability.

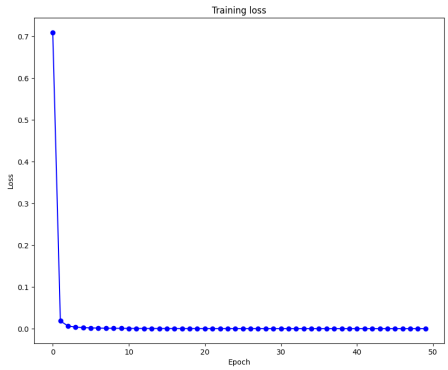

Figure 3: Training Loss on Error Dataset without Movie Key Words

Unlike with the RandomForestClassifier, we were only able to train and test BERT on the more robust error dataset with key movie words using one specific error rate. This is because each run for each error rate takes about an hour, using a costly GPU, so resources are limited. We decided to use a 50% error rate as a sweet spot because this is a little after where the RandomForestClassifier was no longer able to achieve 100% accuracy. Thus, this error rate would give some room for BERT to improve. On this 50% error rate data with moves titles, BERT achieved an accuracy of 99.7%, which is really good. We graphed the training loss for each epoch below, and it decreased to minimum of around 0.000047. The training time was 55 minutes on a T4 GPU.

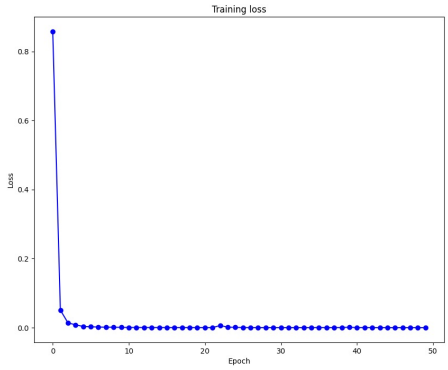

Figure 4: Training Loss on Error Dataset with Movie Key Words

5.6.3 Random Forest Classifier vs. BERT

BERT performed better than the Random Forest Classifier on all datasets. The resilience of BERT against increasing levels of noise is noteworthy, as it not only maintains a high accuracy far longer than the RandomForestClassifier but also showcases the robustness of its pretraining methodology when it comes to understanding and filtering out irrelevant information. This ability to discern and retain valuable data points becomes crucial in practical applications where noise is an inevitable part

of the input data. Furthermore, the sharp decline in accuracy observed in the RandomForestClassifier beyond the 70% noise threshold underscores the limitations of traditional machine learning models in the face of substantial data corruption. It highlights the need for more sophisticated approaches, like BERT, that can leverage context and semantics to withstand high noise levels.

6 Analysis

We know that Enterprise data is messy. That is one of the reasons most of this data go underutilized or unused. This article (3) from Forrester sums it well: On average, between 60% and 73% of all data within an enterprise goes unused for analytics. We propose to use language models to to learn from past tedious and manual mapping activity that enterprises have done and apply that for all future classification work. This fine-tuned model could be used by various organizations.

Business activity data in enterprises have details such as vendor name, cost of activity, timing of it, that can give lot more clues about activity. When business activities data have errors, those additional parameters associated with core data can provide additional context. That is the additional insight got solidified when we looked at data and performance of various models closely. As BERT's ability to pre-tune itself from the data in additional columns such as vendor name, comments, etc, gives it a leg over other models. As a result, BERT far exceeded our expectations, on this simple but error-prone task of classification to improve quality of carbon reporting for organizations.

7 Conclusion

Based on the qualitative analysis, it's evident that both the BERT and RandomForestClassifier models are robust, maintaining accuracy with up to a 60% error rate in the synthetic business activity data. To harness this model's capabilities in practical scenarios, further efforts are needed to adapt it for real-world data application. We believe the optimized hyperparameters should be directly applicable in real-life situations, although fine-tuning them based on actual data observations might require minimal effort. Such refinements are anticipated to enhance carbon emission tracking substantially, enabling organizations to more effectively concentrate on reducing carbon emissions—a critical and urgent goal for humanity.

References

A Appendix (optional)

1) EPA Center for Corporate Climate Leadership: https://www.epa.gov/climateleadership
2) Supply chain emission estimation using LLMs: https://arxiv.org/pdf/2308.01741.pdf
3) Forrester: Hadoop Is Data's Darling For A Reason: https://www.forrester.com/blogs/hadoop-is-datas-darling-for-a-reason/
4) Code: https://github.com/AmarGadkari/cs224n/tree/main