

A NOVEL RANKED EMISSION-FACTOR RETRIEVAL FOR GREENHOUSE GAS EMISSION CALCULATION

Paskaran Sathees

(IT19052748)

BSc (Hons) in Information Technology
Specialising in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2022

A NOVEL RANKED EMISSION-FACTOR RETRIEVAL FOR GREENHOUSE GAS EMISSION CALCULATION

Paskaran Sathees

(IT19052748)

Dissertation Submitted in Partial Fulfillment of the Requirements for the BSc (Hons)
in Information Technology Specialising in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2022

Declaration

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or institute of higher learning, and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to the Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic, or other mediums. I retain the right to use this content in whole or part in future works (such as article or books).

Signature: 

Date: 09/09/2022

The above candidate has carried out this research for the undergraduate dissertation under my supervision.

Signature of the supervisor:

Date:

Abstract

Emission Factors (EF) selection is vital in the Carbon Management Systems (CMS) emission calculation process. Due to Carbon footprint reduction regulations, there is a demand increase for CMS with better usability and scalability. However, most CMS assumes users know emission technologies well. To circumvent these problems, we have proposed a novel EF ranking system with a combined scoring approach. We have considered each EF as a document unit and emission activity information provided by the user as the search query. This combined scoring approach uses a linear combination of the Vector Space Model (VSM), Natural Language Processing (NLP) Word Embedding techniques, and optional time series personalization with Hierarchical Recurrent Neural Networks (HRNN) to rank EF documents for exact and non-exact EF search queries. Using the user satisfaction metric Mean Average Precision (MAP) optimization with the Gaussian process, selected the most suitable word embedding (“glove-wiki-gigaword-300”) and linear combination parameter (0.41). Furthermore, this study discusses performance metrics such as query speed, future EFs scalability, and system resource utilization of this system. This scoring approach improved by nearly 30% and 127% of user satisfaction over VSM and word embedding, respectively. The results conclude that this approach can provide better usability while being scalable for EF selection tasks.

Keywords: Emission Factors, Information Retrieval, Natural Language Processing, Word Embedding, Vector Space Model.

Acknowledgment

I would like to express my sincere gratitude to all those who helped me accomplish my dissertation, especially my supervisors, Ms. Anjalie Gamage and Ms. Sanjeevi Chandrasiri of the faculty of computing, Sri Lanka Institute of Information Technology. Moreover, I would like to thank the panel members Prof. Koliya Pulasinghe, Mr. Vishan Danura Jayasinghearachchi, and Ms. Hasarangi Dhananjana Withanage, faculty of computing, Sri Lanka Institute of Information Technology, for their valuable comments. In addition, I thank my external supervisor Dr. Daniel N Subramaniam, faculty of engineering, university of Jaffna, for the practical domain knowledge.

Table of Contents

Declaration	iii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
List of Abbreviations	xi
List of Equations	xii
1. INTRODUCTION	1
1.1. Background and Literature Survey	1
1.2. Problem & Hypothesis	4
1.3. Research Gap	6
1.3.1. Emission calculator or CMS studies	6
1.3.2. Word embedding studies	7
1.4. Scientific Contribution	9
2. RESEARCH PROBLEM	10
3. RESEARCH OBJECTIVES	13
3.1. Main Objectives	13
3.2. Specific Objectives	13
4. METHODOLOGY	14
4.1. Preliminaries	14
4.1.1. Vector space model (VSM)	14
4.1.2. Word embedding	16
4.2. Complete System Architecture	19
4.3. Overall Scoring Framework	20
4.4. Data Collection	21
4.5. System Architecture	21
4.5.1. Overall EF ranking system architecture	22
4.5.2. Pre-computations	23
4.5.3. Query processing	27

4.5.4.	Personalized re-ranking	29
4.6.	Technologies and Implementation	30
4.6.1.	EF ranking module implementation	30
4.6.2.	Application backend implementation	33
4.6.3.	Application frontend implementation	36
4.7.	Experimentation Methodology	38
4.7.1.	Experimentation criteria	38
4.7.2.	User satisfaction measurement	38
4.7.3.	Query speed measurement	40
4.7.4.	Emission-factor scalability measurement	40
4.7.5.	System resource utilization measurement	41
4.8.	Commercialization	41
5.	RESULTS AND DISCUSSION	45
5.1.	Results	45
5.2.	Research Findings	50
5.3.	Discussion	51
6.	CONCLUSIONS	53
	REFERENCES	54
	APPENDICES	58
	Appendix A. Emission Factor Retrieval Tests	58
	Appendix B. Emission Calculation Tests	59
	Appendix C. Emission Flow UI Prototypes	61
	Appendix D. Sample MAP evaluation dataset	62

List of Figures

Figure 4.1: Visualization of a word vector (“glove-wiki-gigaword-300”) for terms related to traveling mediums.	16
Figure 4.2: Visualization of a word vector (“glove-wiki-gigaword-300”) for terms related to vehicle power sources.	17
Figure 4.3: Visualization of a word vector (“glove-wiki-gigaword-300”) for location-related terms. .	18
Figure 4.4: Complete system architecture with all research components. The components illustrated in color are for this thesis.	19
Figure 4.5: The overall scoring framework used in calculating final ranking scores. It includes general steps (1-4), exact query scoring (A1-A3), and non-exact query scoring (B1-B5). Final scores differ by type of query.	20
Figure 4.6: EF retrieval system architecture. There are two groups (sub-systems) of components: pre-computation (pre-computation 1 and 2) and the query processor.	22
Figure 4.7: Sample normalized standard EF document from DEFRA.	24
Figure 4.8: Sample all term lists.	25
Figure 4.9: Sample custom inverted indexes.	25
Figure 4.10: Structure of length normalized TF-IDF matrix.	26
Figure 4.11: Operations performed during combined score calculation. The Combined score is the linear combination of VSM and embedding score.	27
Figure 4.12: Personalization framework	29
Figure 4.13: Scoring module folder structure (Backend)	31
Figure 4.14: Scoring module folder structure (Research)	31
Figure 4.15: Application database Entity Relation (ER) diagram.	34
Figure 4.16: Data warehouse physical diagram.	35
Figure 4.17: Backend Django project file structure	35
Figure 4.18: Implemented fronted mobile screens 1 and 2 of emission calculation flow	36
Figure 4.19: Implemented fronted mobile screens 3, 4, and 5 of emission calculation flow	37
Figure 4.20: Business model canvas	42
Figure 4.21: Pricing plan	42
Figure 4.22: Product landing page.	43
Figure 4.23: Product promotion pamphlet.	43
Figure 5.1: Coarse MAP results for word vectors and delta values	46
Figure 5.3: Optimization results visualization for user satisfaction using MAP.	47
Figure 5.2: Fine MAP results for word vectors and delta values	47
Figure 5.5: Average CPU time with document count	48
Figure 5.4: Average CPU time with term count.	48
Figure 5.6: Average CPU time with word vector and delta values	49
Figure 0.1: EF retrieval API endpoint’s sample request (Postman)	58

Figure 0.2: EF retrieval API endpoint’s sample response (Postman)..... 58

Figure 0.3: Emission calculation API endpoint’s sample request (Postman)..... 59

Figure 0.4: Emission calculation API endpoint’s sample response (Postman) 60

Figure 0.5: Frontend UI prototype for emission calculation and EF retrieval 61

List of Tables

Table 1.1: A sample EF record from DEFRA 2021 EF Standard	3
Table 1.2: Fields used in CRIS, DEFRA, EPA, IPCC, and NGA EF standard dataset.....	4
Table 1.3: Novelty of the work compared to other emission calculator tools.....	6
Table 1.4: Novelty of the work compared to the application of the same technologies	8
Table 4.1: EF data preparation tasks and affected fields.....	23
Table 4.2: EF ranking module implementation technologies and usage.....	33
Table 4.3: EF ranking and emission calculation's backend implementation technologies and usage ..	36
Table 4.4: Mobile frontend development technologies and usage	37
Table 5.1: Summarized coarse MAP results for word vectors and delta values	45
Table 5.2: Fine MAP results for word vectors and delta values for best word vectors.....	46
Table 5.3: Average query time for vast EF datasets.....	49
Table 5.4: Memory and storage usage by TF-IDF matrices.....	49
Table 0.1: Three Sample MAP evaluation queries and relevant EF documents	62

List of Abbreviations

Abbreviation	Description
NDC	Nationally Determined Contribution
CMS	Carbon Management Systems
GHG	Greenhouse Gases
CO ₂ -eq	CO ₂ -equivalent
GWP	Global Warming Potential
EF	Emission Factor
UOM'	Unit of Measure
CRIS	Climate Registry Information System
DEFRA	Department for Environment, Food and Rural Affairs
EPA	Environmental Protection Agency
IPCC	Intergovernmental Panel on Climate Change
NGA	National Greenhouse Accounts
UK	United Kingdom
VSM	Vector Space Model
IR	Information Retrieval
TF-IDF	Term Frequency-Inverse Document Frequency
TF	Term Frequency
IDF	Inverse Document Frequency
PCA	Principal Component Analysis
BI	Business Intelligence
OCR	Optical Character Recognition
MS	Microsoft
AWS	Amazon Web Services
S3	Simple Storage Service
API	Application Programming Interface
ER	Entity Relation
UI	User Interface
GB	GigaByte
MAP	Mean Average Precision
MRR	Mean Reciprocal Rank
SEO	Search Engine Optimization
MB	MegaByte
CPU	Central Processing Unit

List of Equations

(1) Emission factor calculation.....	2
(2) TF-IDF calculation	15
(3) VSM cosine calculation.....	15
(4) Word embedding cosine calculation.....	17
(5) TF-IDF weight calculation.....	27
(6) Combined score calculation.....	29
(7) MAP calculation	39
(8) AP calculation.....	39
(9) Precision@k calculation	39

1. INTRODUCTION

1.1. Background and Literature Survey

Greenhouse gas emissions (here onwards, mentioned as emissions) or carbon emissions from human activities are the leading cause of global warming. Of course, not all human activities contribute to global warming, e.g., walking. However, there are most human tasks that contribute to global warming. These contributing tasks are called Emission Activities, e.g., driving a gasoline car. As a result, global warming causes severe damage to the natural ecosystem and public health.

Over the past decades, nations gathered and signed many international treaties to reduce emissions and mitigate global warming effects. Among those, the Paris agreement of 2015 is significant as it provided a concrete temperature goal to limit the global average temperature increase below two degrees Celsius. In addition, it required nations to provide their updated carbon footprint reduction measures called Nationally Determined Contribution (NDC) every five years [1], [2]. For example, Sri Lanka provided its NDC in 2021 with its measures for reducing emissions [3]. Governments primarily target corporate organizations or businesses when implementing emission reduction policies because of the significance of their emission contribution to overall emissions. These policies pressures organizations to manage their emissions and take emission reduction measures [1], [4].

These corporate carbon management practices include carbon reporting to measure their emission contribution, emission-reducing efforts, and investing in carbon reduction efforts [4]–[7]. However, recent studies state that the results of these practices are far below the Paris agreement goal's expectations [1]. Earlier corporate carbon management policies have reduced emissions [6], [8]. However, there are still some deficiencies in the practices used by the businesses that restrict the decision-making ability to reduce emissions, e.g., carbon reporting delay can cause a delay in making timely reduction measures [4], [6], [8]. In addition, organizations use emission calculation and Carbon Management Systems (CMS) for their carbon reporting purposes. Emission calculation tools (or CMS) with better accuracy and usability can

improve the effectiveness of carbon reporting by reducing input hassles. Therefore there is a need for emission calculation tools that provide improved usability while still being accurate enough [4]–[6], [9].

Many Greenhouse Gases (GHG) contribute to global warming released during human activity. These GHGs contribute to emission in different ratios, and a unit emission activity can release GHGs in different ratios. Even though some CMSs try to calculate for all these GHG variants, it is tedious, and values are not intuitive to make decisions. Therefore, to simplify, most organizations use a standard unit of kg CO₂-equivalent (CO₂-eq) for emission calculation than measuring each GHG in kg [7], [9]–[11].

Emission calculation is the process of estimating the amount of GHG emitted during human activity [8], [10]–[12]. Emission calculations can occur at several levels, such as personal, product, organizational, city, and country [7], [8]. This thesis only focuses on calculating emissions that occur at the organizational level. The amount of emission for each activity depends on the emission technology (e.g., generator) and the resource consumption amount (e.g., 5-liter diesel). The Global Warming Potential (GWP) is the emission technology's impact on emission [5], [7], [9], [11], [13]. The Emission Factor (EF) captures GWP, and EF estimates how much an emission activity adds to global warming [4], [12]. Therefore, as shown in (1), emission for emission activity of a known EF value can be calculated by multiplying the consumption amount with the EF value [8]–[10].

$$GHG\ Emission = Consumption \times Emission\ Factor\ Value \quad (1)$$

As seen in (1), the selected EF significantly affects the emission calculation's outcome. An EF's value depends on various factors, such as the reaction's chemical equation, chemical compositions of fuels, reaction efficiency, machinery, temperature, vendor, period, and location [4], [6]. A single EF tries to capture all these affecting factors. Therefore, numerous EF observations are needed to denote different variations of these factors. Furthermore, EF values must be updated periodically to maintain their validity because the external factors affecting EF values may change over time [12]. Table 1.1

shows a sample EF taken from the 2021 DEFRA (Department for Environment, Food and Rural Affairs). ‘Scope’ [9], ‘level 1’, ‘level 2’, ‘level 3’, ‘level 4’, and ‘column text’ fields capture factors affecting the EF. ‘UOM’ (Unit of Measure) is the EF’s consumption unit, and ‘GHG’ is the gas type. Finally, the “GHG conversion factor 2021” is the EF value. E.g., the GHG emission for ten passengers traveling 1km in an airplane with this EF would be 1.5102 kg CO₂-eq.

Table 1.1: A sample EF record from DEFRA 2021 EF Standard

Dataset Field	Value
Scope	Scope 3
Level 1	Business travel- air
Level 2	Flights
Level 3	Short-haul, to/from UK
Level 4	Economy class
Column text	With RF
UOM	passenger.km
GHG	kg CO ₂ e
GHG conversion factor 2021	0.15102

The service provider or the manufacturer can provide the best EF value for the emission technology. However, most organizations use EF published by recognized government and private entities for use within their geographical boundary. These entities estimate EF values and publish them periodically for the organization’s usage. Organizations can select the most suitable EF publisher (also called EF standard). These published EF are called default EF and cover most emission activities. There is a high observable variation in fields and structures used by EF standards. Table 1.2 shows fields used by popular emission factor standards such as CRIS (Climate Registry Information System), DEFRA (Department for Environment, Food and Rural Affairs), EPA (Environmental Protection Agency), IPCC (Intergovernmental Panel on Climate Change), and NGA (National Greenhouse Accounts). Organizations can use these default EF in case of missing a manufacturer or service provider given technology-specific EF [4], [8]. Most of these default EF datasets are for usage within a selection of countries, e.g., DEFRA is for the UK [8]. If the country-specific EF dataset is unavailable, organizations can use an international standard like the IPCC

standard [6]. This thesis only considers default EF. However, it is extendable for custom EF.

Table 1.2: Fields used in CRIS, DEFRA, EPA, IPCC, and NGA EF standard dataset

EF Standards	Year	Dataset Fields
CRIS	2021	Fuel type, Heat content, Carbon content, Fraction oxidized, CO2 emission factor (Per Unit Energy), CO2 emission factor (Per Unit Mass or Volume), Unit
DEFRA	2021	Scope, Level 1, Level 2, Level 3, Level 4, Column text, UOM, GHG, GHG conversion factor
EPA	2021	Fuel type, Heat content (HHV), CO2 factor, CH4 factor, N2O factor, CO2 factor, CH4 factor, N2O factor, Unit
IPCC	2006	EF ID, IPCC 1996 Source/Sink Category, IPCC 2006 Source/Sink Category, Gas, Fuel 1996, Fuel 2006, Type of parameter, Description, Technologies, Parameters, Region, Abatement, Other properties, Value, Unit, Equation, IPCC Worksheet, Technical reference, Source of data, Data provider
NGA	2020	Fuel combusted, Energy content factor, Emission factor

There are many emission calculators or CMSs to accomplish corporate and public carbon calculation and carbon reporting [13]. Emission calculators use EF to calculate emissions, and responsibility for selecting an appropriate EF relies on the user trying to calculate or store emissions. From the observations, most emission calculators provide two different interface types for EF selection during emission data input.

1. Groups of EF are formed by categorizing EF, e.g., food, transportation, and residential [13], [15], [16].
2. Dropdowns with all EF [14].

1.2. Problem & Hypothesis

Studies evaluating emission calculators' user satisfaction [14]–[16] have recommended a higher demand for better usability to keep users engaged in reporting their emissions and increase user retention. However, most of the EF selection

interfaces provided by these emission calculators have issues that reduce usability and scalability. For example, using EF groups formed by categorizing EF can reduce accuracy and limits the number of emission types the calculator can support. Furthermore, it can get cluttered and unusable if it tries to support more emission types. Therefore, the EF group interface is considered less scalable and restrictive for business use cases.

On the other hand, dropdown interfaces can scale to any number of EF. However, it adds complication and inefficiencies to the users. Moreover, it can become tedious since most users have to enter emissions frequently within a short time [14].

The proposal proposed this novel EF ranking system to solve the issues with emission calculators' EF selection interface by improving the usability and scalability of the emission calculator EF selection interfaces. This proposed system ranks EF using the emission activity data given by the user as a query. It assumes the user will be able to explain their emission activity within queries, e.g., "Drove to work using diesel-powered Toyota Prius," "Travelled from the UK to India with a first-class airplane ticket," or "Stayed at a Turkish hotel." This proposed system should be able to rank and recommend EF for these kinds of non-exact free-text queries. The system considers each EF record as a document unit. The system expects the internal labeling of the EF to be a black box for users, and users do not need any prior knowledge about EF. However, users should have some level of understanding of the emission technologies they are using. Therefore, there is a high chance that user queries to be non-exact and not include exact wordings as EF labels. In this case, any exact search approach that matches exact characters will result in a poor ranking outcome. The proposed system uses a combined scoring approach of combining the word embedding technique with a Vector Space Model (VSM) to overcome this issue [17]. Since most users use the same emission technologies or tools in their routine life, the system also provides a personalization option to re-rank emission factors according to their previous emission histories.

Hypothesis: The combined rank approach will improve usability and be scalable for the EF selection process.



Experimentations and evaluations were conducted with the implemented EF ranking system and measured its usability and scalability metrics to test the hypothesis. In addition, the evaluation procedure compares several publicly available and custom-trained word embeddings. Finally, this thesis provides insights on selecting the most appropriate word embedding for the EF ranking application and discusses the achievement of the combined scoring approach compared to the IR model or word embedding.

1.3. Research Gap

Due to the novelty of this thesis, there are no studies of emission calculators with this kind of EF ranking features. This research is the first to combine ranking features with the EF selection task during emission calculation. However, there are studies on emission calculators or CMS and ranking technology used in this thesis implementation.

1.3.1. Emission calculator or CMS studies

Table 1.3: Novelty of the work compared to other emission calculator tools

Works	Emission Factor Searching	Personalization
The state of carbon footprint calculators: An evaluation of calculator design and user interaction features [18]	X	X
Mobile-Based Carbon Footprint Calculation: Insights from a Usability Study [14]	X	X
Development of a Web Application for Individual Carbon Footprint Calculation [15]	X	X
A novel approach to calculate individuals' carbon footprints using financial transaction data – App development and design [16]	X	X
This work		

There are numerous emission calculators for individual and corporate usage; comparing all is an overwhelming task. Therefore, as shown in Table 1.3, this thesis compares these studies [14]–[16] and this review study [18] that evaluated 31 online emission calculators. In addition, the thesis compares EF selection interfaces, other ways of getting emission records, and suggestions from previous studies.

As previously explained, most emission calculators either use drop downs [14] or group selection [15], [16] for their EF selection interfaces. These dropdowns and groups are associated with EF at the software level. However, these kinds of software-level associations are less scalable. This web application [15] uses categories to associate EF, e.g., food and electricity.

This novel research [16] implements a mobile application that uses individuals' financial records for emission calculation. However, even though it is sufficient for individual use cases, it is difficult to differentiate personal and business financial records in a business environment. Moreover, it might be morally tricky for most organizations to provide their financial records due to their sensitivity.

This study [14] implemented a mobile emission calculator for personal usage and conducted a user study on the implemented system. The results of this user study state that most users find the emission calculation approach too complex and needs improvement. This review study [18] that compared over 30 online emission calculators also states that the usability and user-friendliness of these emission calculation tools need improvements.

1.3.2. Word embedding studies

This thesis uses the word embedding technique with VSM to achieve the best EF retrieval outcomes. Previously numerous studies have used word embedding in Information Retrieval (IR) tasks within other domains, such as software engineering and biomedicine (as shown in Table 1.4).

Table 1.4: Novelty of the work compared to the application of the same technologies

Works	VSM	Word Embedding	Parameter Tuning	Personalization Re-Ranking
Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports [17]	✓	✓	X	X
Recommending Similar Bug Reports: A Novel Approach Using Document Embedding Model [19]	✓	✓	X	X
A comparison of word embeddings for the biomedical natural language processing [20]	X	✓	X	X
This work	✓	✓	✓	✓

This study [17] has used word embeddings to recommend similar software bugs and has successfully improved results. The study has also used bug report structural information for the ranking. However, this structural information is unsuitable in EF retrieval due to structural differences between EF standards. This work [19] further improves bug recommendation by using document embedding. These bug recommendation studies have used combined scores only to rank; however, in this thesis, ranking switches between VSM and combined scores depending on the query types. Finally, this study [20] discusses an attempt at biomedical retrieval with no successful improvements. However, it has only used word embedding scores for ranking and has not combined word embedding scores with other IR models.

1.4. Scientific Contribution

This thesis provides three main contributions to the research community and implementor of such systems.

1. A novel EF selection approach

This thesis details this novel EF ranking system using the combined scoring approach. The combined scoring approach uses IR model (in this case, VSM) scores and word embedding scores to find a final ranking score. The EF ranking system ranks EF documents with these final ranking scores.

2. Designing and implementation details of this approach

The METHODOLOGY chapter elaborates on how such an EF ranking system could be designed architecturally and implemented to achieve the same results as this thesis. The thesis also shows the tools and technologies used in implementing this approach.

3. Evaluation criteria and metrics for this approach

This thesis states the criteria and associated metrics when evaluating this kind of EF ranking approach. Future studies can use these metrics to compare variations of EF ranking systems for corporate usage with similar ranking approaches.

4. Results and findings of evaluating this approach

The RESULTS AND DISCUSSION chapter provides results and findings from evaluating this novel system according to defined criteria. The thesis also states the most suitable word embedding and parameter range for experimentation embedding models. Furthermore, it provides insights and reasoning behind the results and findings.

5. Parameter optimization and best word vector selection methodology

Work also discusses using optimization algorithms to find the best-ranking parameter values and selecting the best possible word vectors using the user satisfaction measure.

2. RESEARCH PROBLEM

EF selection during emission calculation is a vital task that users must carry out very carefully since the outcome of the calculated emission value heavily depends on it. Moreover, choosing incorrect EF will ultimately provide inaccurate data for decision-making personnel. Previously most CMS and emission calculators are used by experts or knowledgeable personnel with emission calculation. However, the proposed complete system's (The Carbonis: Real-Time Carbon Neutrality Management and Optimization Using Natural Language Processing) target users are general employees with minimal understanding of carbon reporting. Therefore, previous EF selection methodologies can be overwhelming and tedious for the target user group while carrying out EF selection carefully.

Implementing an efficient and comfortable EF selection mechanism within the emission calculation tools is necessary. The Background and Literature Survey indicates that there are mainly two types of interfaces provided by previous emission calculators: groups and dropdowns. The thesis tries to solve this issue by providing an EF search interface for EF selection. In the proposed complete system, user input for emission calculation is a free-text representation of the user's emission activity. Therefore, the ranking system should rank EF by considering this free-text emission activity as the search query. Therefore, providing ranked EF for EF selection can reduce the user's effort in finding matching EF.

The user does not need to know about internal EF labeling or representation. Therefore, most queries (emission activity data) are free-text queries and mostly will not overlap with the internal EF labeling. However, the EF ranking system should provide the best-ranking outcome for free-text (non-exact) and exact queries.

EF standard publishers publish EF datasets in various document formats and styles. In addition, EF standard publishers publish updated datasets periodically (mostly yearly). Therefore, emissions should use the relevant year's EF, even if the latest EF is available [21]. Usually, these emission technologies are represented in these

documents using technical terms different from what a general employee might call them. Therefore, the EF ranking system should quickly adopt these EF standard changes and hide all these variations from the user minimizing complexity.

IR searches and ranks documents to fulfill user needs from the user-given query. There are various IR models developed over the years, e.g., VSM. These IR models can provide superior effectiveness for exact matching occasions where query terms somewhat match the document terms. However, for the EF ranking system, plane IR models are insufficient because of the free-text queries. The EF ranking approach uses a word embedding model to solve issues with the free-text queries. Word embedding provides similarity scores of terms. Even though word embedding alone is enough to rank EF, the performance of plane IR models is way better than word embedding for exact queries. Therefore, the EF ranking system uses a combined ranking approach by combining both scores to provide the best of both methods.

Research question: Will the combined ranking approach improve usability while being scalable for the EF selection process?

Implemented EF ranking system will take queries with emission technology (e.g., vehicle, generator, electricity.) and other emission calculation values, emission activity's year, and reporting standard for the organization as the inputs. In addition, the EF ranking system will provide ranked EF as the output.

This thesis assumes the following assumptions while implementing this novel EF ranking system.

1. Users can describe emission activity in plain text for the query

For thesis implementation, the EF ranking system only supports query input given in English text format. This assumption is a limitation for most organizations due to language and disability barriers. However, a future extension could add multilanguage support and other forms of input, e.g., voice.

2. User given query will include the necessary details

This EF ranking system assumes the query inputs are accurate enough and relevant to the emission activity the user is trying to report. Moreover, these queries are free from spelling errors (in the complete system, the initial component should clean for errors before providing a query for EF ranking).

3. An organization will only use a single EF standard for reporting

Usually, an organization will only use a single EF standard at a time. The EF ranking system will only provide EF from an EF standard set by the employee's organization.

4. Organizations will only add emissions for default EF

There may be occasions where a much more appropriate EF value is available for the emission activity than the ones in the default EF standards, e.g., manufacturer-given EF. However, this implementation only supports emissions for these default EF and does not support using custom EF. Future implementation could extend to support custom EF when adding emissions.

This thesis implements three functional requirements related to EF ranking: EF ranking for emission activity query, optional personalized EF ranking, and calculating emission (discussed within app development). In addition, the optional personalization provides additional personalized ranking by keeping track of user history. This EF ranking component should provide better usability and scalability as stated in the hypothesis, e.g., scalable with most EF standards.

3. RESEARCH OBJECTIVES

3.1. Main Objectives

The main objective of this study is to design and implement a ranked EF retrieval system for the EF selection process of emission calculation. This EF ranking system should work with the complete proposed system's cross-platform mobile application to manage organizations' carbon emissions in real-time. This EF ranking approach will also use a combined ranking approach to combine the IR model and word embedding similarity measures to provide results for free-text queries. Finally, the implemented EF ranking system should have better usability and scalability. This EF ranking can optionally use personalization for further customized ranking results.

3.2. Specific Objectives

For the achievement of this work's primary objective of implementing an EF ranking retrieval that would provide ranked EF results using the combined ranking approach for the user query and parameters; the work should also realize the following specific objectives,

1. Collection and pre-processing of EF standard datasets

Initially, acquire datasets of popular EF standards from suitable sources and pre-process with data cleaning and necessary data transformations.

2. Creation and storage of EF in standard representations

Design a standard EF representation for the document unit of the EF ranking system. This standard EF representation should accommodate popular EF standards. Finally, store pre-processed EF in this standard EF representation.

3. Implement a ranked EF retrieval system to rank EF document units.

Design and implement the EF ranking module with the combined ranking approach, which uses similarity, term frequency, and optionally personalization to rank EF results.

4. App development related to emission calculation

Design and implement an emission calculation module to calculate emissions for the provided emission activity data and selected EF. Warehouse these records for further analysis and reporting.

4. METHODOLOGY

This chapter discusses the implementation details of the EF retrieval system with the combined rank approach. First, the Preliminaries section reviews the core concepts used in this work, such as Vector Space Model (VSM) and word embedding. Next, the Complete System Architecture section shows where this thesis work fits with the complete proposed system (Carbonis). Then the Overall Scoring Framework section explains how the scoring algorithm computes ranking scores for the combined-ranking approach. Next, the Data Collection section discusses the EF dataset collection process. Next, the System Architecture section illustrates the structure of the EF ranking system components. Next, the Technologies and Implementation section explain the development technologies and techniques used for this work. Then the Experimentation Methodology section showcases the setup used to measure the metrics needed to test the hypothesis. Finally, the Commercialization section explains the business and marketing strategies taken to market this EF retrieval system.

4.1. Preliminaries

This section gives a theoretical review of the core concepts used for the EF ranking system. This thesis implementation uses VSM for the IR model. Therefore, this section briefly explains VSM and word embedding techniques.

4.1.1. Vector space model (VSM)

IR tries to find documents that fulfill user needs from a collection of documents (called text corpus) by using relevancy calculated between user queries and documents [22], [23]. However, all IR models have a bottleneck issue on how users express their needs in queries that impacts IR results. Moreover, an IR system's effectiveness depends on document storage and retrieval effectiveness. That is why document collection usually undergoes processing that changes documents into a uniform format before document storage. Several IR models with high retrieval effectiveness are available, and the popular choices include VSM and probabilistic models [22], [23].

This EF ranking approach uses VSM with TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity due to their wide reputation and solid theoretical basis [17], [22]. TF-IDF provides statistical importance of a keyword for a specific corpus [23], [24]. As shown in (2), TF-IDF is simply a product of Term Frequency (TF) and Inverse Document Frequency (IDF) values. TF represents the number of times the keyword appeared in a document, and IDF indicates the rarity of that keyword in the whole text corpus [17], [22], [24]. Documents with rare terms occurring will have a high TF-IDF value. There are diverse interchangeable TF and IDF [22] formula forms.

$$TF - IDF = TF \times IDF \quad (2)$$

VSM uses vectors to represent document and query terms and linear algebra vector operation operations for calculations [22], [23]. Furthermore, query and document vectors stores query weights and TF-IDF weights calculated according to (2), respectively, and will have a size equal to the total number of unique terms in the corpus. A cosine similarity score provides similarity between query and document vectors. As shown in (3), the cosine score is the dot product of length normalized query vector (\vec{q}) and document vector (\vec{d}) (3) [17], [22]. Cosine score value can take any value ranging from zero to one. Vectors are length-normalized by dividing vector values by vector length (L₂ norm). Vector length normalization provides robust scaling for operations [25], [26]. A higher cosine similarity value means the document vector has high relevancy to the query vector [22].

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} \quad (3)$$

There are many known issues with VSM, TF-IDF, and cosine similarity scorings, such as,

1. Not suitable for large vocabulary due to the copious vector computations [23].
2. Inefficient for continuously expanding corpus due to the complete TF-IDF vector updates needed for a small number of changes.

3. Ignorance of the syntactic meanings [22], [24]
e.g., treats ‘eat’ and ‘ate’ as different words
4. Ignorance of the semantic meanings [22], [24]
e.g., treats ‘vehicle’ and ‘car’ as different words

Issues (1) and (2) are irrelevant to this EF ranking approach because existing EF datasets do not get invalid with the release of a new dataset. Old EF is valid for its service period. Deriving primary word forms using stemming and lemmatization fix the syntactic issue (3) [24]. However, there is still the issue of missing semantic meanings (4). Therefore, there is a necessity to combine a semantic similarity model with VSM to provide semantic understanding for EF retrieval tasks [23].

4.1.2. Word embedding

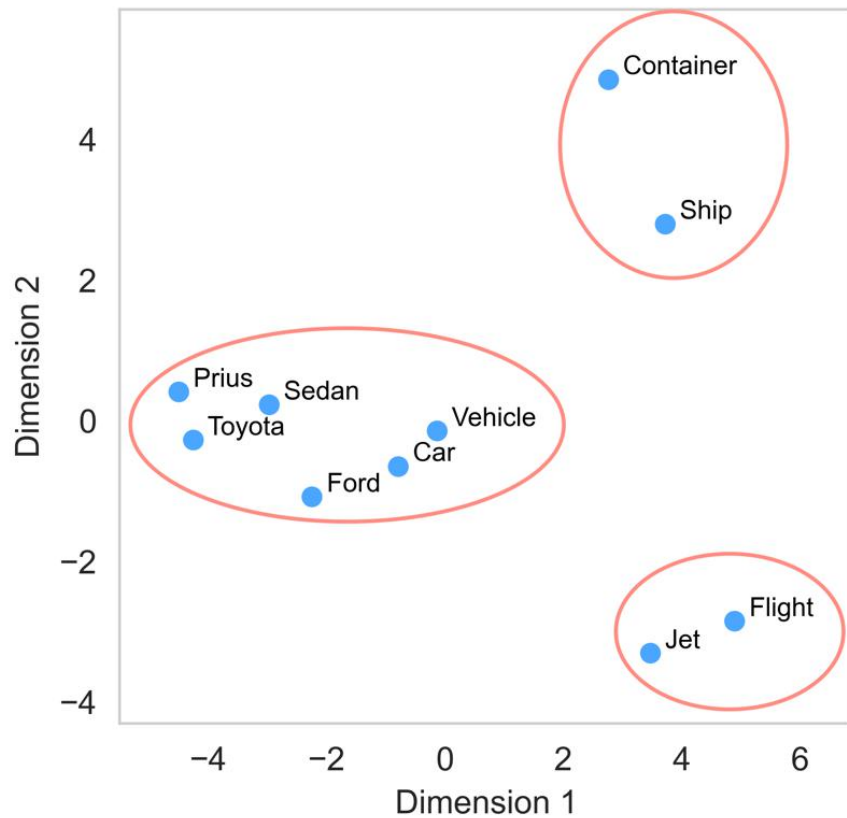


Figure 4.1: Visualization of a word vector (“glove-wiki-gigaword-300”) for terms related to traveling mediums.

The thesis implementation combines word embedding with VSM to provide a semantic understanding. Word embedding assumes that co-occurring words have similar meanings [25], [27], and it captures semantic meanings of words within real number vectors [26]–[28]. Word embeddings tend to perform better than other count models [17]. It is a general practice to train deep learning models on a large text that stores knowledge on word vectors [17], [25]. These word vectors are byproducts of deep learning operations. As shown in (4), the semantic similarity value of a set of terms is the cosine similarity of these terms’ length normalized vectors (\vec{w}). These cosine score values can take any value ranging from zero to one [25].

$$\cos(\vec{w}, \vec{w}) = \vec{w} \cdot \vec{w} \quad (4)$$

Word embedding selection involves careful consideration of the embedding’s corpus domain, corpus size, training algorithm, and output vector dimension. A word

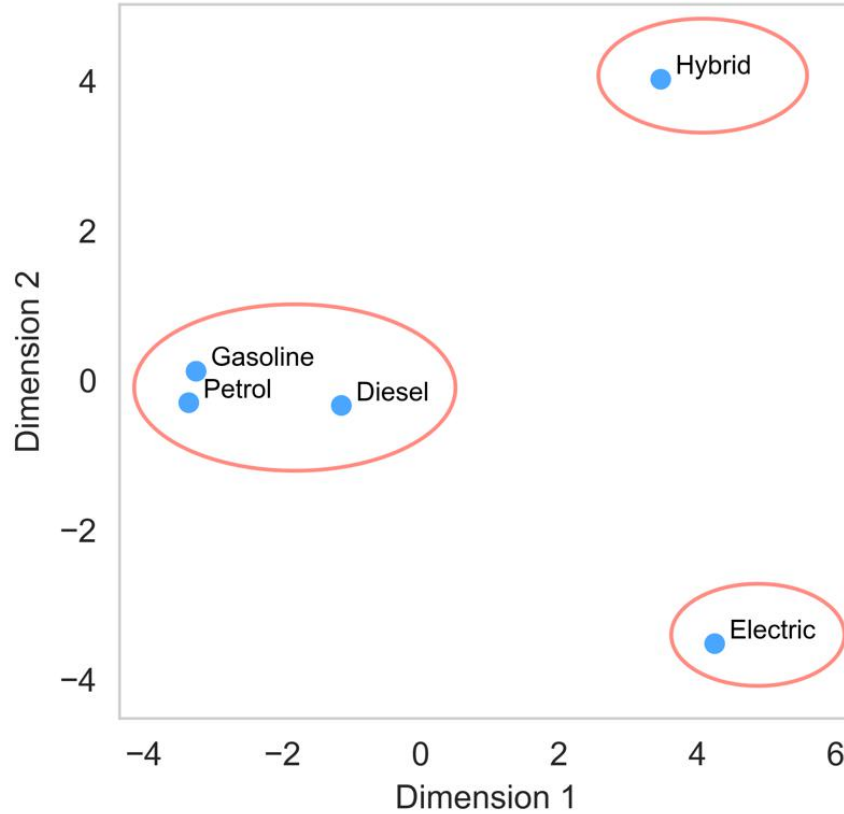


Figure 4.2: Visualization of a word vector (“glove-wiki-gigaword-300”) for terms related to vehicle power sources.

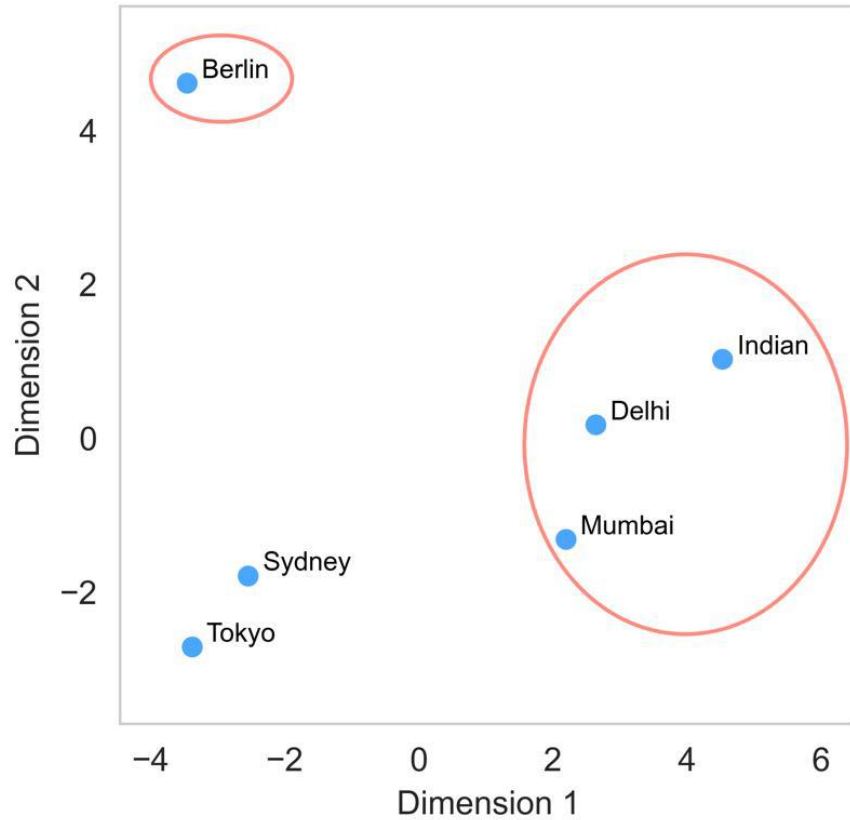


Figure 4.3: Visualization of a word vector ("glove-wiki-gigaword-300") for location-related terms.

embedding with a corpus of high relevance to the application field of decent size can generate better results than a huge corpus with an irrelevant domain. However, a larger same-domain corpus can provide better results [26], [27]. Word embedding training algorithms includes word2vec [29], GloVe [30], FastText [31], [32], ConceptNet [33], and so on [26]. Neural network algorithms provide better accuracy and efficiency than knowledge-based algorithms [28], and in most cases, GloVe provides better results [26]. Standard word vector dimensions range from 50 to 300, and like corpus size, higher word vector dimensions can improve results [26], [27]. Word vectors are a better choice for IR tasks because word vectors usually utilize low system resources compared to trained models. Word vectors support huge vocabulary because of their huge training corpus compared to the application corpus (in this case, the EF dataset). However, there may be situations where word vectors may not include some terms and will need error handling for those cases. This issue occurs because the English

language has a much larger vocabulary than the training corpus will ever have, and it is improbable to find a domain-related corpus that supports all English vocabulary.

Due to the lower EF dataset vocabulary size, there is a significant probability for user query terms to be absent in the whole EF dataset. Therefore, this EF ranking approach could provide results for most queries by taking advantage of the sizeable token space of word vectors. Figure 4.1, Figure 4.2, and Figure 4.3 show how similar context words appear closer than others. Here, the dimension of word vectors has been reduced to two dimensions using Principal Component Analysis (PCA) for visualization. The axes of these graphs represent these two arbitrary dimensions. Therefore, word embedding can provide context understanding useful for EF ranking applications.

4.2. Complete System Architecture

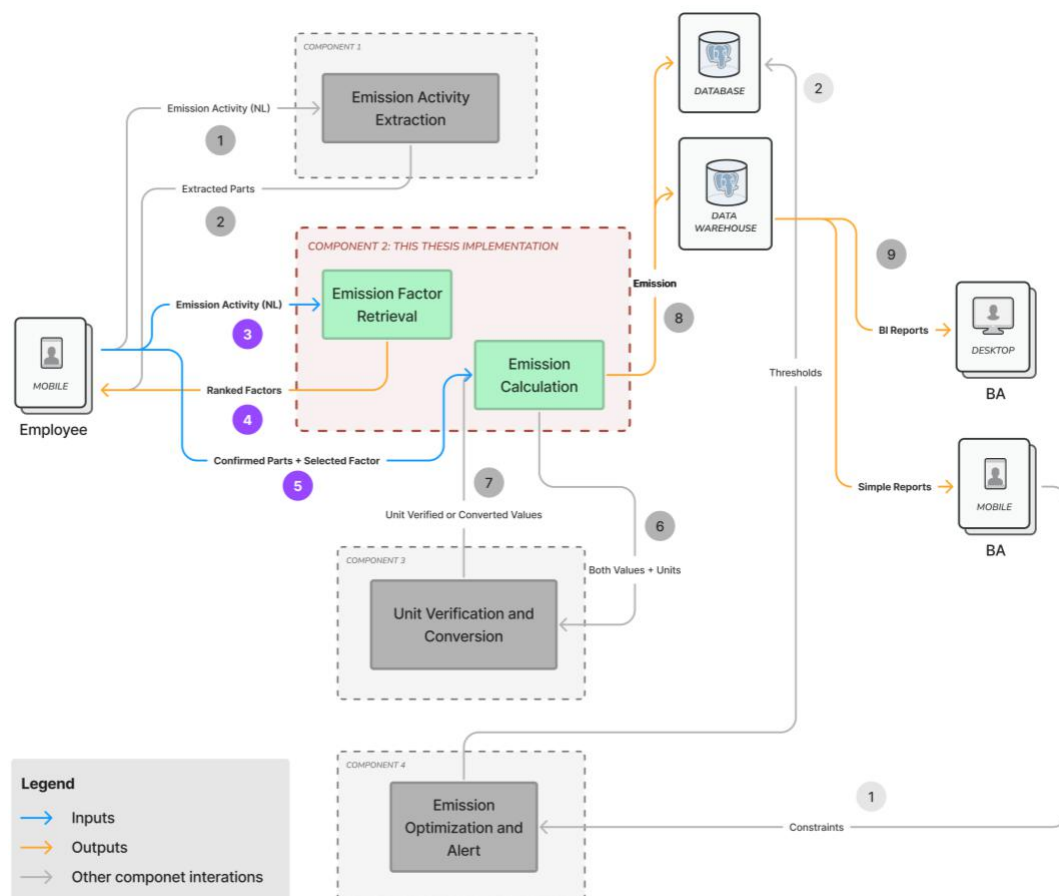


Figure 4.4: Complete system architecture with all research components. The components illustrated in color are for this thesis.

As shown in Figure 4.4, this thesis is for component 2 of all four components of the proposed complete system architecture. Component 2 included two main parts: EF ranked retrieval and emission calculation. EF ranked retrieval takes natural language (in this case, English) emission activity data (query) as input along with activity year value extracted by component 1 and EF standard set by the organization. Users provide queries and view the ranked EF result using a mobile application. After confirming parts extraction (from component 1) and selecting suitable EF from the ranked results, the system will calculate emission and store it in the database and data warehouse. Emission values are sent to component 3 to verify and convert units during emission calculation. Business users can generate Business Intelligence (BI) reports from mobile or off-the-shelf BI tools with the emission data stored in the data warehouse.

4.3. Overall Scoring Framework

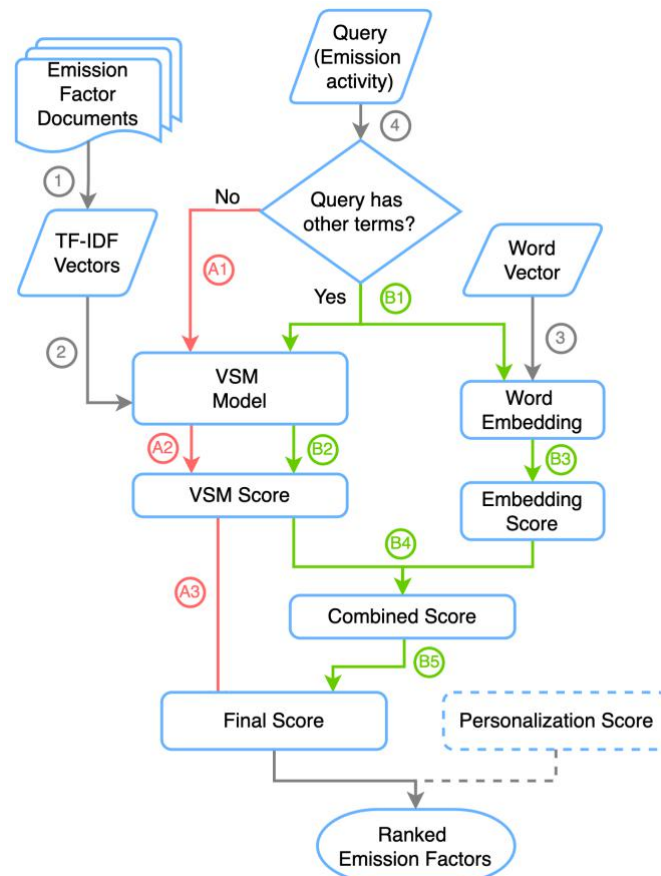


Figure 4.5: The overall scoring framework used in calculating final ranking scores. It includes general steps (1-4), exact query scoring (A1-A3), and non-exact query scoring (B1-B5). Final scores differ by type of query.

The scoring framework accommodates exact and non-exact EF queries as the retrieval approach accepts free-text queries. Most IR models like VSM could effectively rank for the corpus vocabulary (index terms) and will not work for other terms (non-index terms). As explained earlier, a lower vocabulary count (DEFRA 2014 to 2021 corpora altogether have only 401 terms) increases the chances of non-exact queries. Therefore, the retrieval system should rank EF results for both query types. Figure 4.5 shows the EF retrieval system scoring framework, using a combined ranking approach to calculate a final score. Initially, the preparation tasks, such as creating TF-IDF vectors from EF documents (stored in a standard document format after pre-processing), loading TF-IDF, and loading word vectors, will happen. Once the query is received, it will check for non-index terms' presence, and if it is an exact query, it will only calculate the VSM score, which will be the final score. Otherwise (non-exact query), it will separate the query terms into the index and non-index terms and calculate the VSM score and embedding score respectfully. Next, it will derive the final combined score from these scores for the non-exact query. By doing so, the scoring framework will be able to rank exact and non-exact EF queries using the best possible scoring. Finally, if there is personalization is enabled, an additional personalization score is added to the final ranking score.

4.4. Data Collection

Before implementing the EF retrieval system, raw EF datasets for default EF standards (shown in Table 1.2) from 2014 to 2021 have downloaded from the publishers' official websites. These EF datasets were available in different formats (e.g., PDF, Excel, Word) and structures (e.g., DEFRA has a full, concise, and automatic processing set). Furthermore, the observable EF dataset update frequency is at least a year. Finally, for convenience, the DEFRA automatic processing dataset has been chosen for this thesis's initial implementation. However, the design and development of the EF system have considered all these five EF standards (shown in Table 1.2).

4.5. System Architecture

This section gives a practical point of view on how components of the EF ranked retrieval system communicate to achieve the final ranked results. First, this section

presents an overall system architecture with the required components for EF ranking. Then, moving on, it discusses sub-systems in much detail. Finally, this section shows how optional personalization occurs for the EF ranking.

4.5.1. Overall EF ranking system architecture

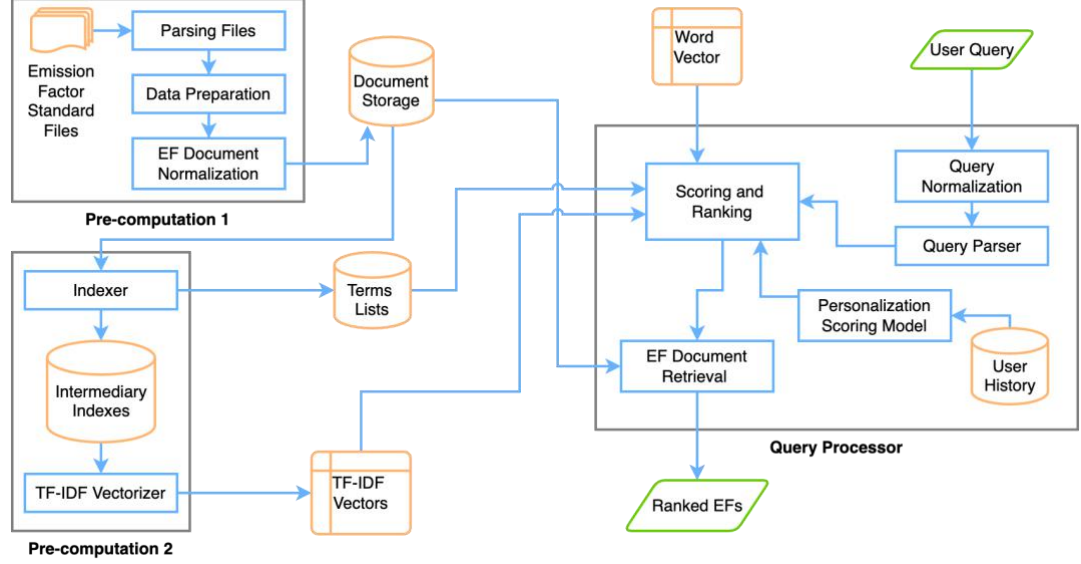


Figure 4.6: EF retrieval system architecture. There are two groups (sub-systems) of components: pre-computation (pre-computation 1 and 2) and the query processor.

As shown in Figure 4.6, the EF ranking system architecture includes preparation and query processing components. Mainly there are two sub-systems, such as pre-computation and query processor. The pre-computations components are only for usage during the application development and when updating for a new EF dataset. Pre-computation involves manual tasks using scripts. Pre-computation takes raw EF datasets as input and stores cleaned EF documents, terms lists, intermediary indexes, and TF-IDF vectors as outputs. On the other hand, the query processor requires an EF standard (set by company policy) and year of emission activity alongside the emission activity query and will output ranked EF documents.

It is enough to provide EF ranking results for a single EF standard-year filter combo. Therefore, during precomputing terms lists, intermediary index and TF-IDF vectors were created for EF standard-year combo (e.g., DEFRA 2021 and DEFRA 2020).

Then, the query processor will use appropriate terms lists, TF-IDF vectors, and word vectors during query processing.

4.5.2. Pre-computations

Table 4.1: EF data preparation tasks and affected fields

Data Preparation Tasks	Affected Fields
Remove unnecessary fields to reduce data size	ID and discontinued fields
Rename field names	All fields
Add identification fields	EF standard and year
Change fields' data type	Fields with incorrect data type
Remove other GHG records to reduce the data size	Records other than CO ₂ equivalent GHG
Generate text field for EF document corpus	"Level 1", "Level 2", "Level 3", "Level 4", and "Column text"
Remove unnecessary records to fix data quality issues	duplicated, empty, and missing EF values rows

Figure 4.6 shows two distinguished pre-computation groups: pre-computation-1 (PC1) and pre-computation-2 (PC2). This distinguishing is done according to the alteration need of pre-computation tasks to adopt new EF standard datasets, i.e., PC1 might need some alteration, and PC2 would not need any. PC1 converts raw EF datasets and saves normalized EF documents (in standard document format) on a database by executing the following three main tasks sequentially.

1. Parsing EF dataset files

First, raw data (DEFRA automatic processing Excel dataset) were loaded and parsed into manipulatable software structures. Optical Character Recognition (OCR) or manual conversion can extract data from file types that are harder to parse (e.g., PDF and MS word).

2. Data preparation

Applying data wrangling and transformation operations (shown in Table 4.1) to the parsed EF data ultimately generated text fields and formed clean EF documents.



Figure 4.7: Sample normalized standard EF document from DEFRA

3. EF document normalization

During EF document normalization, tokens from the text field were created by applying necessary language processes and stored normalized EF documents with their tokens in a database. These language processing tasks were,

- Cleaning text field by removing blank lines and replacing contractions
- Word tokenization: slicing text strings into token lists
- Punctuation marks removal
- Changing to lowercase
- Stop word removal: stop words are the frequently occurring and less contributing words, e.g., “a” and “the”
- Lemmatization: finds dictionary root word forms of nouns and verbs. PC1 uses lemmatization over stemming (finding roots by trimming the rear parts of words) because it provides meaningful root forms, unlike stemming. Moreover, most word embeddings use lemmatization during training.

```

> 1996: Array
> 2006: Array
  _id: ObjectId('62e2c2c0f8c694328d3e77f3')
  emission_standard: "IPCC"

> 2014: Array
~ 2015: Array
  0: "africa"
  1: "aggregate"
  2: "air"
  3: "also"
  4: "aluminium"
  5: "america"
  6: "anaerobic"
  7: "arabia"
  8: "artics"
  9: "articulate"
  10: "asbestos"

```

Figure 4.8: Sample all term lists

```

> 2014: Object
> 2015: Object
> 2016: Object
> 2017: Object
> 2018: Object
> 2019: Object
> 2020: Object
> 2021: Object
  _id: ObjectId('627c0bdaf85fba91f5f2e3c8')
  emission_standard: "DEFRA"

~ 1996: Object
  ~ aa: Object
    doc_freq: 2
    ~ docs: Array
      ~ 0: Object
        doc_id: 29641
        term_freq: 1
      ~ 1: Object
        doc_id: 29642
        term_freq: 1
    ~ aan: Object
    ~ abandon: Object
    ~ abatement: Object

```

Figure 4.9: Sample custom inverted indexes

Finally, for saving these normalized EF documents (as shown in Figure 4.7), a NoSQL database (MongoDB) was used due to its data model flexibility, which

allows for saving normalized EF documents of different EF standards in a single collection.

PC2 uses normalized EF documents stored and creates terms lists and TF-IDF vectors by executing the following three main tasks sequentially.

1. Indexing

Indexer creates separate intermediate indexes for EF standard-year pair. The search index is not mandatory for query processing. However, using it to achieve programming simplicity and efficiency during the TF-IDF matrix creation. The indexer creates a custom inverted index (shown in Figure 4.9) with document frequency (df_t) and term frequency ($tf_{t,d}$) and stores it in the NoSQL database. As shown in Figure 4.8, the indexer stores a list of all terms in a separate database collection.

	Doc ID 1	Doc ID 2	Doc ID 3
Term 1	0	0	0
Term 2	0	0.256	0.512
Term 3	0.456	0	0
Term 4	0	0.368	0
Term 5	0	0	0

Length Normalized TF-IDF Matrix

Figure 4.10: Structure of length normalized TF-IDF matrix

2. TF-IDF matrix creation

Using intermediary indexes for each EF standard-year pair, the TF-IDF vectorizer creates TF-IDF matrices (Figure 4.10). These TF-IDF matrices will have dimensions: the number of terms and the number of EF document IDs. Furthermore, the TF-IDF matrix's weights for the term-EF document pair ($W_{t,d}$) calculation uses (5), where N is the total number of EF documents within

an index. Finally, TF-IDF vectorizer stores created TF-IDF matrices as files to the local file system.

$$W_{t,d} = \left(1 + \log_{10}(tf_{t,d})\right) \times \log_{10}(N/df_t) \quad (5)$$

3. Obtaining or training word vectors

Many pre-trained word vectors are publicly available with generous licenses. It is also possible to create custom-trained word vectors from a corpus. In this case, the thesis used 11 pre-trained word vectors and a custom-trained word2vec word vector for comparisons. The training corpus needed for this custom-trained word vector training resulted from web-scraping of Wikipedia pages for the terms in the DEFRA intermediate index.

4.5.3. Query processing

Query processor ranks normalized EF documents for user-given queries using query essentials (or pre-computation outputs): terms lists, TF-IDF matrices, and word vectors). Before accepting search queries, the query processor will persist query

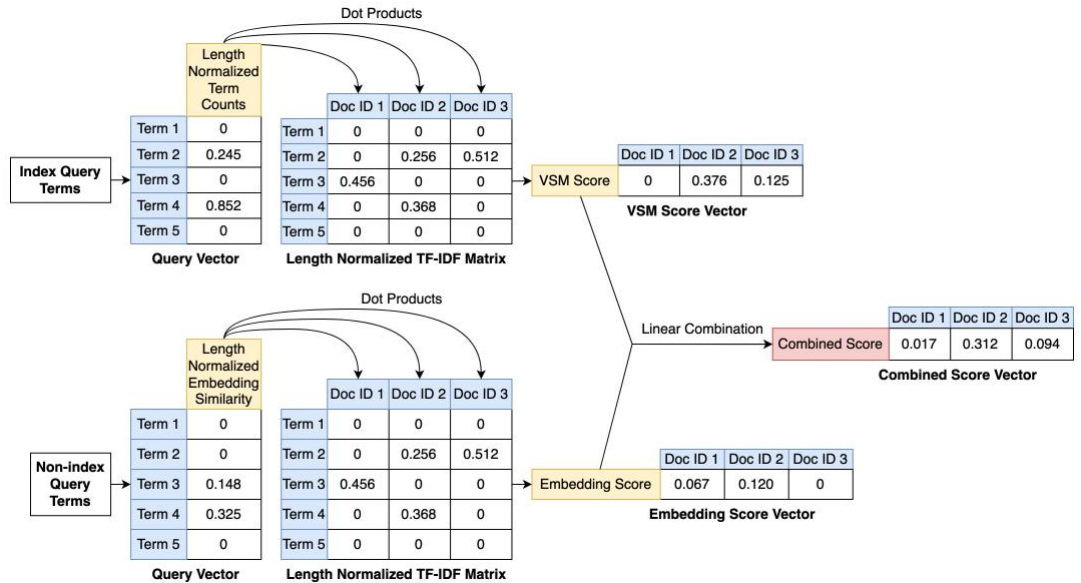


Figure 4.11: Operations performed during combined score calculation. The Combined score is the linear combination of VSM and embedding score.

essentials on memory to reduce overhead. During query processing, the query processor does the following four main tasks.

1. Query normalization

Like EF document normalization, the query processor will apply the same normalization operations on the query string literal to get a list of query tokens.

2. Query parsing

The query parser will check query tokens for non-index term's presence, i.e., exact or non-exact query. Then, the parser will forward query tokens to score and rank either with the VSM score only (exact query) or combined score (non-exact query).

3. Scoring and ranking

The query processor will calculate and rank either with the VSM score or combined scores (Figure 4.11) as guided by the query parser (Figure 4.6).

- a. VSM scoring

VSM scoring only uses term lists and the TF-IDF matrices. Initially, the query processor will create a count query vector from query tokens, which will be length normalized. This length-normalized query vector will have a dimension equal to the number of terms. Then the query processor will create a VSM score vector (dot products of length normalized query and TF-IDF vectors) (Figure 4.11's top part). The sorted VSM score vector will result in EF ranking order.

- b. Combined scoring

Combined scoring additionally uses word vectors than VSM scoring. First, the query processor will create separate query vectors for index (only VSM, same as (a)) and non-index terms. Non-index query vectors will have embedding scores from the word vector and will be length normalized. Embedding score is the similarity between non-index terms and each term of all terms list. Next, the query processor will create an embedding score vector by computing dot products of length normalized non-index query vector and TF-IDF vectors (Figure 4.11) and will create a combined score vector (CS_d) by applying the linear combination formula with VSM score vector (VS_d) and embedding score vector (ES_d). Here, linear combination

value delta (δ) will control the word embedding scoring impact in ranking. Delta's value can range from zero (only VSM ranking) and one (only embedding ranking). The sorted, combined score vector will result in EF document ranking order.

$$CS_d = (1 - \delta) \times VS_d + \delta \times ES_d \quad (6)$$

4. EF documents retrieval

Finally, the query processor will fetch chunks of normalized EF documents from the database according to the sorted EF document ID order and provide them to the user.

4.5.4. Personalized re-ranking

Figure 4.12 shows the optional personalization ranking framework. If the user has enabled the personalization feature in their profiles, the EF ranking system will re-rank the top five hundred EF documents ranked using a combined ranking approach. For

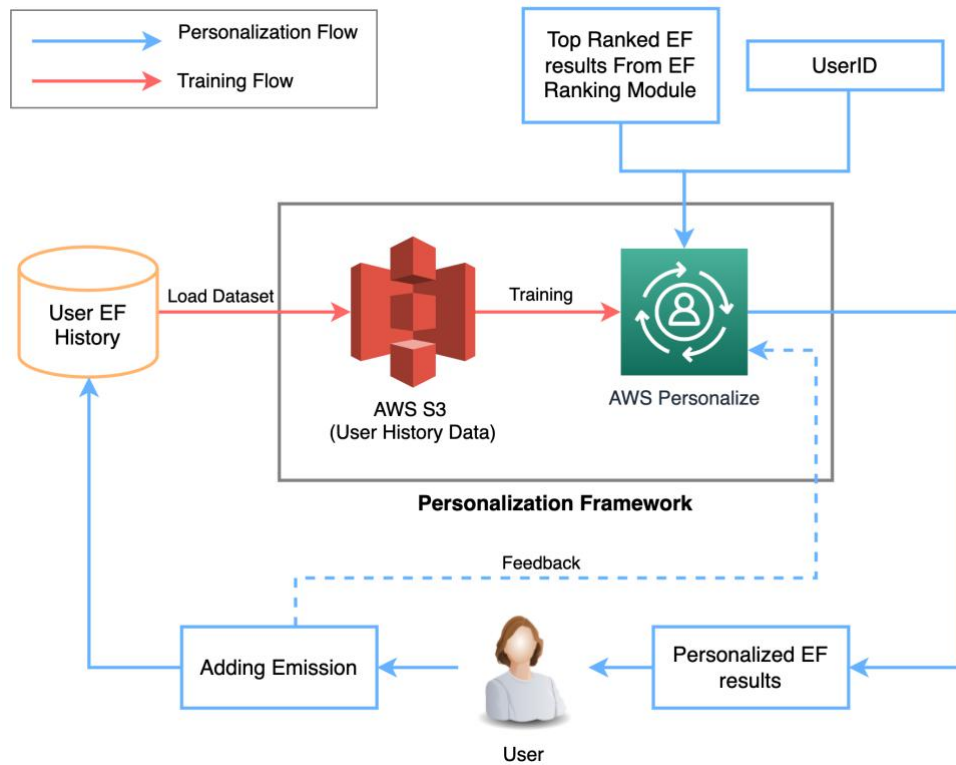


Figure 4.12: Personalization framework

this purpose, personalization utilizes user EF selection histories of the past emission activity data. User EF selection history includes the following four fields,

1. User ID: Unique identifier for the user
2. EF ID: Unique identifier for the EF document
3. Timestamp: Timestamp for the emission record
4. Event Type: Type of event, e.g., selection, impression
5. Event Value: Whether the event occurred (1) or not (0)

Periodically, users' EF selection histories stored in the application database are transferred to Amazon Web Services (AWS) Simple Storage Service (S3) buckets as the personalization training dataset. AWS personalize gives an API endpoint for personalization ranking after training using this data. This endpoint will respond with personalization ranking scores for the top five hundred EF results from combined ranking and user ID. Personalization training happens only periodically manually to reduce unwanted expenses. However, a feedback mechanism will receive the most recent user behavior. The feedback mechanism will send the latest EF selection history when adding a new emission.

4.6. Technologies and Implementation

This section discusses the technologies and tools used to implement this EF ranking system. First, this section shows the scoring and ranking module implementation. Then, it discusses the application's backend and frontend development in terms of EF ranking and emission calculation. Finally, this section shows details on the personalized EF ranking development technologies.

4.6.1. EF ranking module implementation

Figure 4.14 shows the folder structure for EF ranking research and development. The EF ranking module is a collection of python files with the functions needed for EF ranking. Some functions do import functions from other python files. Then these functions were used in a Jupyter environment during research to evaluate the functionalities and test the results. Therefore, some of these files are only for the research environment. Figure 4.13 show the folder structure of the EF ranking module

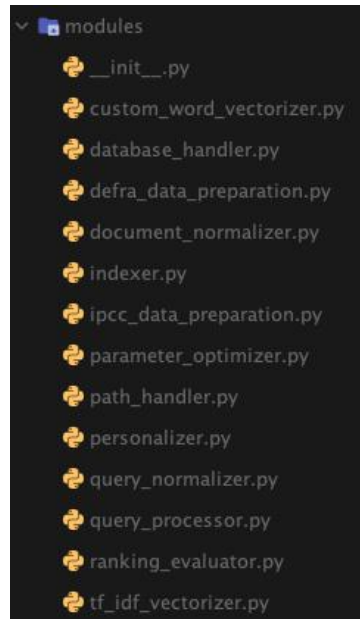


Figure 4.14: Scoring module folder structure (Research)

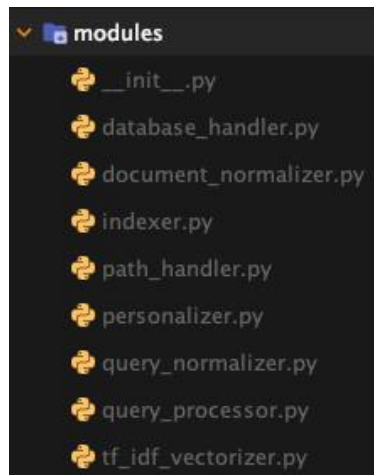


Figure 4.13: Scoring module folder structure (Backend)

in the backend application. The backend application's Application Programming Interface (API) endpoint request will call a single function of the query_processor.py file with necessary parameters (query, EF standard, year, and delta value) to rank EF documents. Further personalization will happen if the user has enabled the personalization feature; else endpoint will return EF documents in the ranked order. Table 4.2 shows the tools and technologies used during the research and development of the EF ranking system.

AWS personalization implementation included the following tasks,

1. User history dataset gathering

A user history dataset is compulsory for the model's training for personalization ranking. Therefore, this dataset will be available in the system database in a production system. However, since this system is still developing, this work has used a synthetic dataset.

2. Pre-processing dataset

Pre-processed user history dataset to match the correct input required by the AWS personalize, e.g., converting timestamp to Unix timestamp.

3. Creating personalization dataset group and schema

AWS personalize explicitly needs a dataset group and schema before building a solution.

4. Configuring AWS S3 bucket and importing dataset to the bucket

The training dataset will be in an AWS S3 bucket, and necessary access to this bucket was given to the AWS personalize.

5. Training an AWS personalize solution

Using this dataset, trained an AWS personalize model with the hyperparameter optimization method and chose the best solution by comparing the metric report given at the end of solution building. The cost will occur for personalization training.

6. Creating campaigns and endpoints

A campaign needs to be created for the best solution to make inferences. In the end, it will give an endpoint to make inferences. Campaigns will incur an hourly cost depending on the number of inferences made.

7. Integration with the query processor

This endpoint is integrated with the query processor and will provide personalization scores for the top 500 EF documents. After adding the personalization score to the IR score, EF documents are re-ranked.

8. Creating an event tracker and integrating it with the system

To make inferences fresh with the latest data, AWS personalize provides a feedback mechanism called Event Tracking. This work integrates it with the application and will send new data to the AWS personalize while a new

emission activity is added to the system by the user. However, it is still necessary to train again after a certain period with up-to-date data.

Table 4.2: EF ranking module implementation technologies and usage

Technology	Usage
Beautiful Soup	<ul style="list-style-type: none"> • Web scrapping Wikipedia pages for custom word embedding training
Boto3	<ul style="list-style-type: none"> • For AWS services
Gensim	<ul style="list-style-type: none"> • Custom word vector pre-processing and training • Obtaining pre-trained word embeddings • Finding embedding similarities
Joblib	<ul style="list-style-type: none"> • Save TF-IDF matrices as files in the file system
Jupyter	<ul style="list-style-type: none"> • Research and testing EF ranking system
MongoDB	<ul style="list-style-type: none"> • NoSQL database for cleaned EF documents, terms lists, and intermediary indexes storage
NLTK	<ul style="list-style-type: none"> • Language processing tasks, e.g., word tokenization, stop word removal, lemmatization
NumPy	<ul style="list-style-type: none"> • Linear algebra and vectorized operations
Pandas	<ul style="list-style-type: none"> • Data preparation • TF-IDF matrix creation • Query scoring • EF ranking
PyCharm	<ul style="list-style-type: none"> • Research and development IDE
Pymongo	<ul style="list-style-type: none"> • Connecting and querying to MongoDB within python
Python	<ul style="list-style-type: none"> • Programming language
Python-dotenv	<ul style="list-style-type: none"> • Managing credentials in a secure way by storing them as environmental variables
Scikit-learn	<ul style="list-style-type: none"> • PCA on word vectors for visualization
Seaborn, matplotlib	<ul style="list-style-type: none"> • Data exploration
SonarLint	<ul style="list-style-type: none"> • Code quality validation

4.6.2. Application backend implementation

Figure 4.15 shows the application database’s Entity Relation (ER) diagram. It highlights relations “Emission” and “User History” in red color, and these highlighted relations are relevant to the EF ranking systems and emission calculation functionality. Figure 4.16 shows the physical diagram of the backend data warehouse model. This

Data warehouse uses a star model. Business users can use this data warehouse to

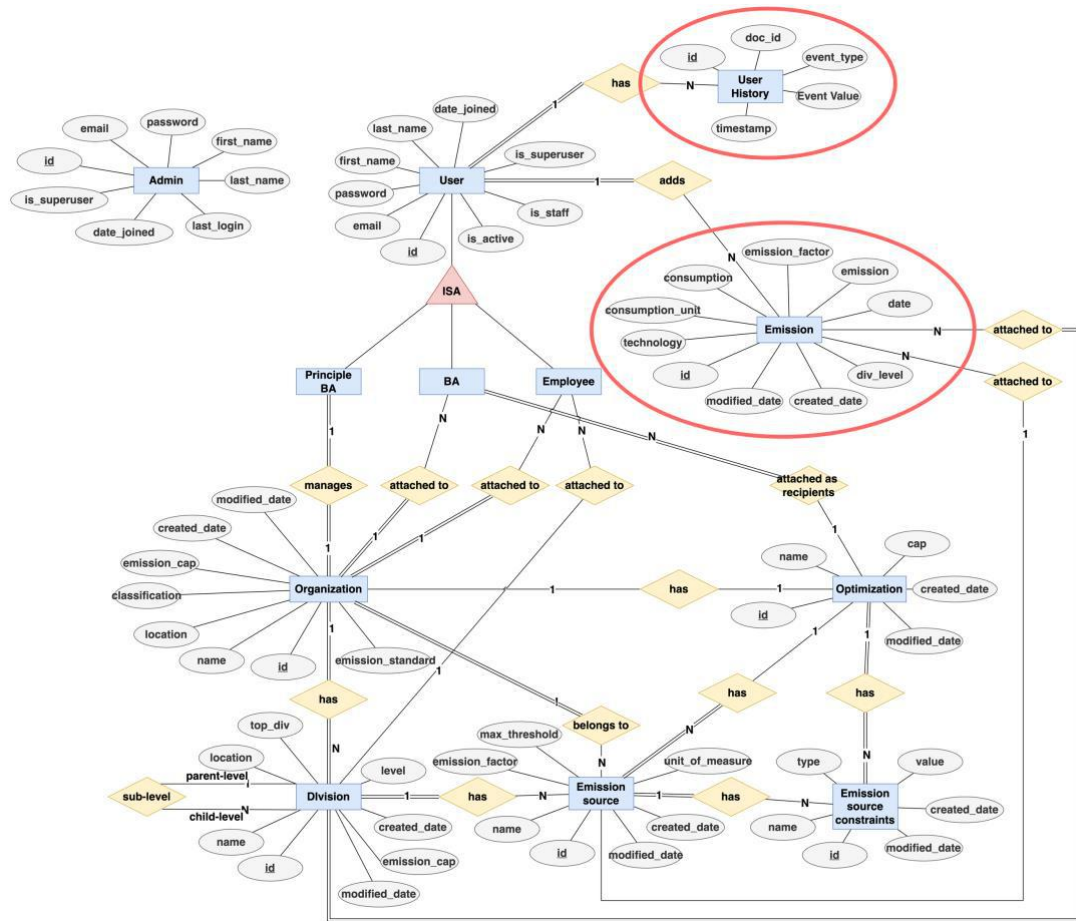


Figure 4.15: Application database Entity Relation (ER) diagram

generate BI reports on their carbon emissions. Figure 4.17 shows the backend Django project folder structure. The “efr” app is responsible for EF ranking features, and the “api” app provides all other features. Django’s AppConfig facility supports the loading and persisting of files on the application startup. Therefore, the app will load and make word vectors, term lists, and TF-IDF available using this facility for EF retrieval API endpoints, thus avoiding overhead loading of these query essentials. API endpoint testing samples for EF retrieval and emission calculation are available in Appendix A. Emission Factor Retrieval Tests and Appendix B. Emission Calculation Tests, respectively. Table 4.3 lists the tools and technologies used during the development of the EF ranking system and emission calculation backend system.

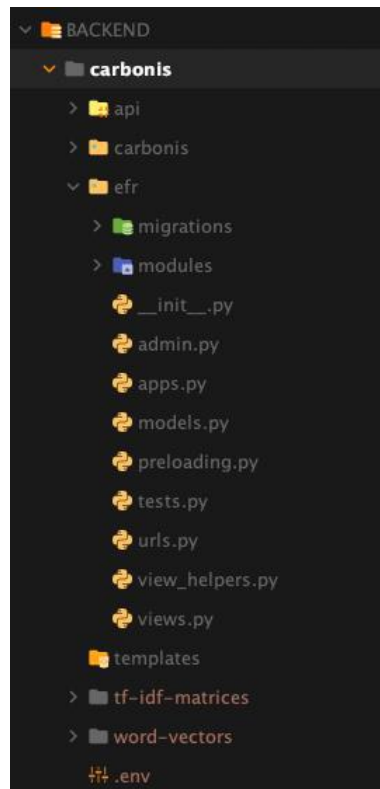


Figure 4.17: Backend Django project file structure

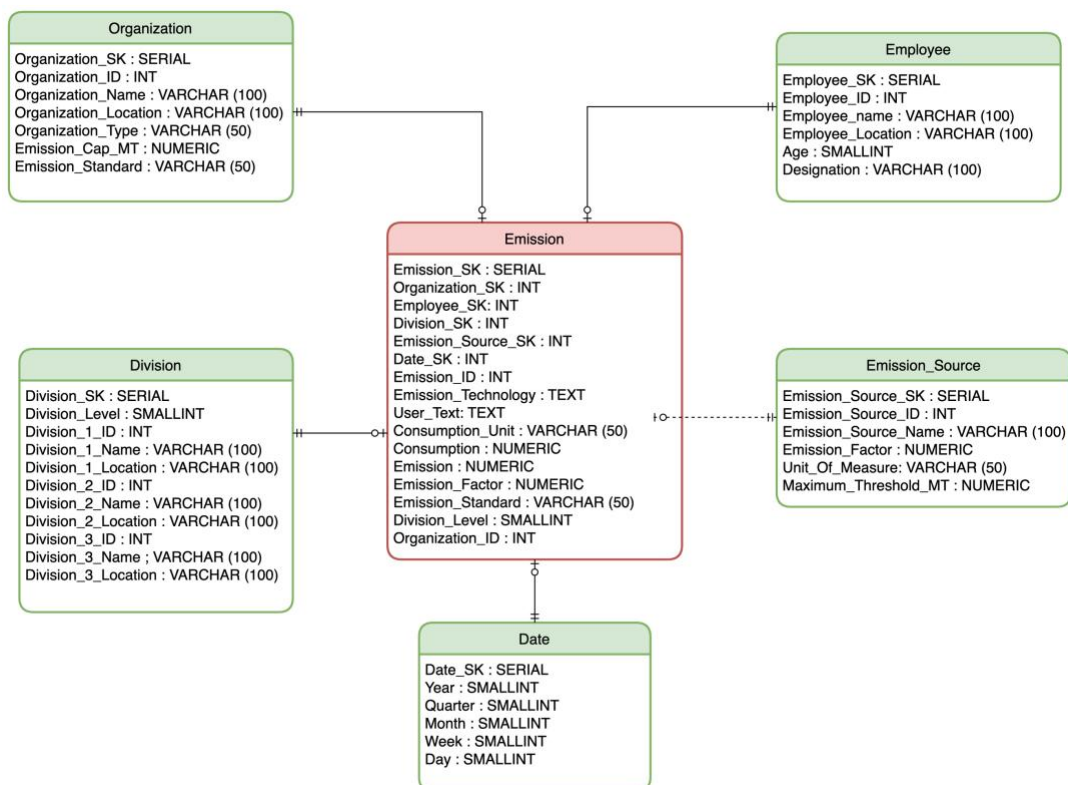


Figure 4.16: Data warehouse physical diagram

Table 4.3: EF ranking and emission calculation's backend implementation technologies and usage

Technology	Usage
Django	<ul style="list-style-type: none"> Backend framework
Django-rest	<ul style="list-style-type: none"> Representational State Transfer (REST) API capabilities
PostgreSQL	<ul style="list-style-type: none"> Application database and data warehouse
Postman	<ul style="list-style-type: none"> API endpoint testing
PyCharm	<ul style="list-style-type: none"> Development IDE
Python	<ul style="list-style-type: none"> Programming language
Python-dotenv	<ul style="list-style-type: none"> Managing credentials in a secure way by storing them as environmental variables
SonarLint	<ul style="list-style-type: none"> Code quality validation

4.6.3. Application frontend implementation

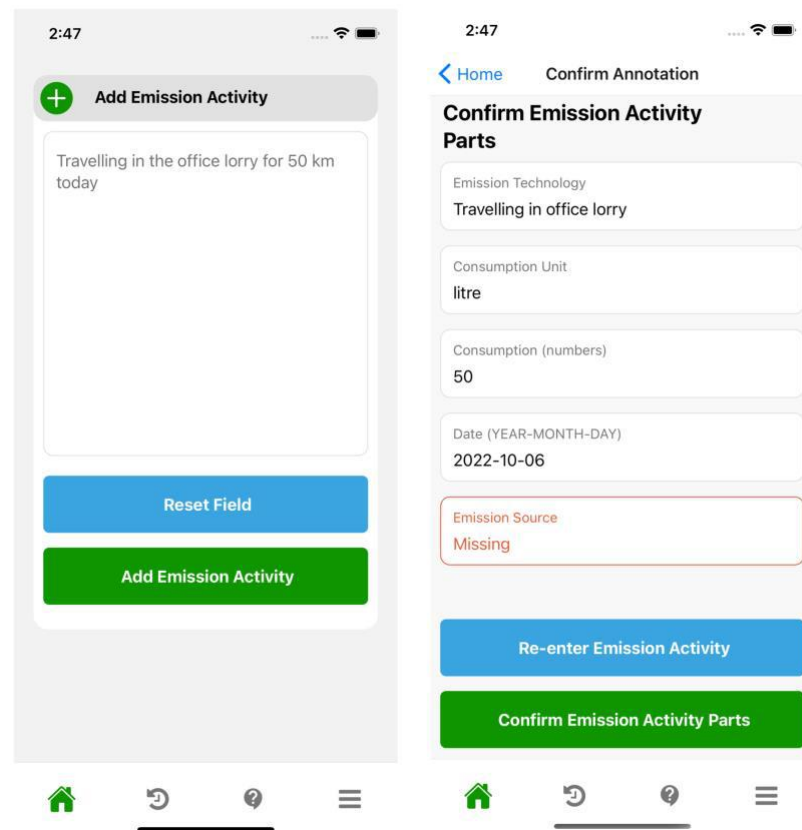


Figure 4.18: Implemented fronted mobile screens 1 and 2 of emission calculation flow

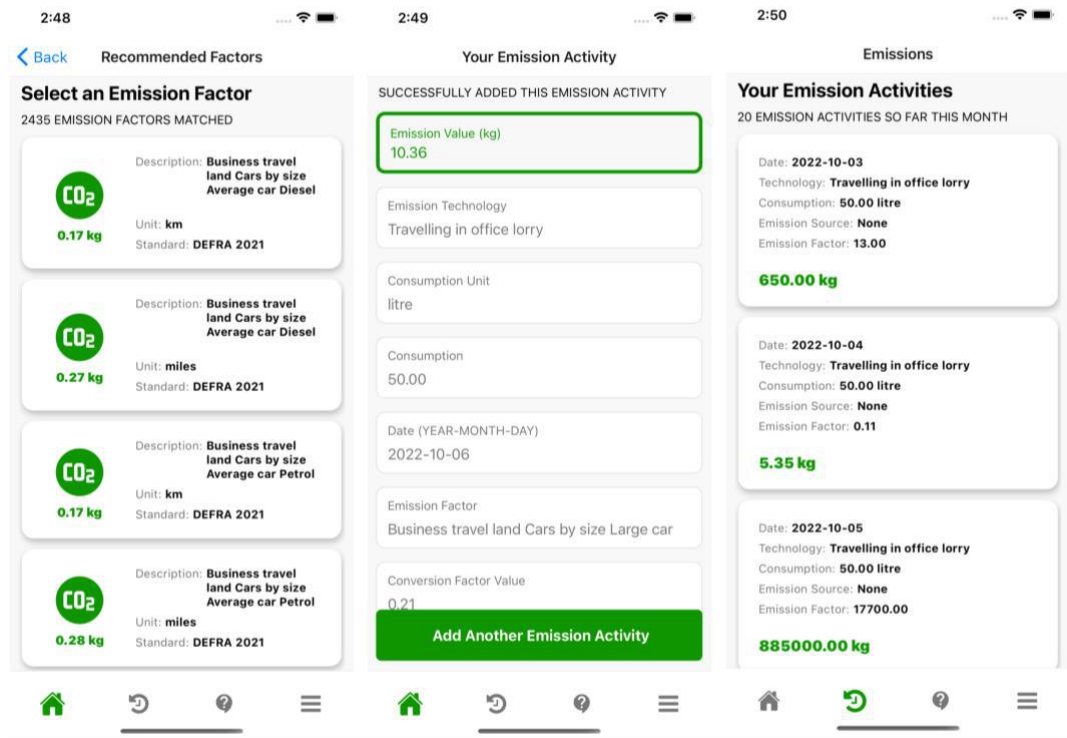


Figure 4.19: Implemented fronted mobile screens 3, 4, and 5 of emission calculation flow

Figure 4.18 and Figure 4.19 shows the frontend mobile screens implemented within the emission calculation flow. Additionally, work implements the frontend applications' authentication, authorization, and navigation parts. Table 4.4 lists the technologies and tools used in mobile application frontend development. In addition, mobile User Interface (UI) prototypes related to emission calculation and EF-ranked retrieval are available in Appendix C. Emission Flow UI Prototypes.

Table 4.4: Mobile frontend development technologies and usage

Technology	Usage
Expo-CLI	<ul style="list-style-type: none"> Representational State Transfer (REST) API capabilities
Figma	<ul style="list-style-type: none"> User Interface (UI) prototyping
JavaScript	<ul style="list-style-type: none"> Programming language
WebStorm	<ul style="list-style-type: none"> Development IDE
React-Native	<ul style="list-style-type: none"> Cross-platform mobile application development
SonarLint	<ul style="list-style-type: none"> Code quality validation

4.7. Experimentation Methodology

This work involved experimenting with the implemented EF ranking system to evaluate the hypothesis and making observations using these experimental results. In addition, due to the novelty of this system, experiments specifically evaluated only this thesis approach and have not compared previous implementations. Therefore, the following sections discuss the experimentation criteria, metrics, and experimental methodologies to measure these evaluation metrics.

All experiment results are from the implemented EF ranking system running in an Apple MacBook Air (M1, 2020) with 8 GB RAM and 512 GB storage running macOS Monterey (version 12.5).

4.7.1. Experimentation criteria

According to the thesis's hypothesis, implementing a novel EF ranking system with the combined ranking approach should have better usability and scalability. Therefore, the experimentation criteria are the EF ranking system's usability and scalability. In addition, user satisfaction and query speed can contribute to the overall usability criteria of the system. Furthermore, EF scalability and system resource utilization can contribute to the overall scalability criteria. Therefore, experimentations should measure four metrics, such as user satisfaction, query speed, EF scalability, and system resource utilization, to test the thesis hypothesis.

4.7.2. User satisfaction measurement

A retrieval system's success heavily depends on the user's satisfaction with the results, and there are several measures available to measure user satisfaction, e.g., precision, recall, F1 score, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and precision@k (P@k) [34]. However, some are specific for ranking evaluation, e.g., MAP, MRR, and P@k.

This thesis uses MAP as the single user satisfaction measure due to its best efficiency for the evaluator effort needed to measure [34]. Therefore, a higher MAP value signifies better user satisfaction. MAP is the mean of the Average Precision (AP) of

all evaluation queries, as shown in (7), where AP is the average precision and q is the number of evaluation queries. AP is the average of Precision@k values, as shown in (8), where k is the document's position in the result and r is the number of relevant query results. Precision@k is the ratio of relevant results at the kth result, as shown in (9), where R_k is the number of relevant documents in top k results.

$$MAP = \frac{\sum_{n=1}^q AP_n}{q} \quad (7)$$

$$AP = \frac{\sum_{n=1}^r Precision@K_n}{r} \quad (8)$$

$$Precision@K = \frac{R_k}{k} \quad (9)$$

A labeled evaluation dataset with evaluation queries and preferred EF documents was needed to measure MAP scores. However, there was no standard evaluation dataset for the EF domain. Therefore, this work used a custom evaluation dataset with 50 evaluation queries (50 provide sufficient statistical significance [34]) and preferred EF documents from a 244 EF document sample (See Appendix D. Sample MAP evaluation dataset). EF document sample was sampled from DEFRA 2021 documents with simple random sampling without replacement to reduce the evaluator's efforts on labeling. The evaluation query set includes queries of different emission topics in a mixture of non-exact and exact queries.

After configuring the EF retrieval system with sample EF documents, measured the following while executing evaluation queries with replacements,

1. MAP value for word vectors (twelve) and delta values.

There was an expectation that MAP values (user satisfaction) to change with the word vector and delta. Therefore, the initial run used delta values between zero and one with 0.05 increments. Then the second run used delta values between 0.35 and 0.5 with 0.01 increments for more refined results.

2. Automated evaluation to find the best word vector and delta values.

Evaluating user satisfaction to find the best word vector and delta value combo can be tedious and exhaustive if analyzing for all possible values of the delta. This experimentation considered using gradient ascend and surrogate models such as the Gaussian process, grid search, and tree-based search to simplify this evaluation process.

4.7.3. Query speed measurement

Query retrieval speed is also substantial for the usability of a search system. Therefore, experiments measured the average query time (average time) to evaluate retrieval speed. A low average time signifies a high query speed, and a high query speed contributes to better usability.

This experiment measured both wall and CPU times in milliseconds. The experiment measured the average speed for the following combinations,

1. Overall average time

The experiment measured it by running the evaluation query set 5 times on DEFRA (2014 to 2021) and IPCC (1996 and 2006) EF documents.

2. Average time with term and document counts

For the term and document counts' effect, the experiment measured the average time running the evaluation query set twice on IPCC EF documents (200 queries) by changing term and document counts.

3. Average time with word vectors and delta values

The experiment measured average time (200 queries) by changing word vectors and delta values on DEFRA documents to validate the word vectors' and delta values' effects.

4.7.4. Emission-factor scalability measurement

Corporate organizations use numerous EF standards. In addition, EF dataset publishers update and release EF datasets frequently. Therefore, the EF retrieval system's scalability significantly depends on its ability to support new EF standards and upcoming datasets with minimal effort.

This experiment measured EF scalability for the following,

1. The effort needed to add support for a new EF standard

At the end of the EF retrieval system implementation, it only supported the DEFRA EF standard. The experiment tried adding support to IPCC (1996 and 2006) to evaluate EF scalability and measured person-hours taken for this adoption task.

4.7.5. System resource utilization measurement

By observing system resource usage, i.e., memory and storage, one can understand the system's scalability to larger user space. Lesser system resource utilization indicates higher scalability for less infrastructure.

This experiment measured system resource utilization for the following,

1. Storage usage by TF-IDF vectors and word vectors

The experiment measured it using file system information.

2. Memory usage by TF-IDF matrices

Only measured memory usage of TF-IDF vectors using Pandas information due to the difficulty measuring exact memory usage by word vectors because Gensim handles it in an optimized manner internally.

4.8. Commercialization

This section discusses the commercialization steps and future marketing strategies to promote this system. Completed commercialization techniques as follows,

1. Market analysis

As stated in the Background and Literature Survey, a market study was conducted for the system (Carbonis) and found a high demand for emission calculation tools with high usability.

2. Creating a business model

Figure 4.20 shows the business model canvas illustrating the system's business model.

3. Creating a pricing plan

Figure 4.21 illustrates the current pricing plans according to the business model canvas.

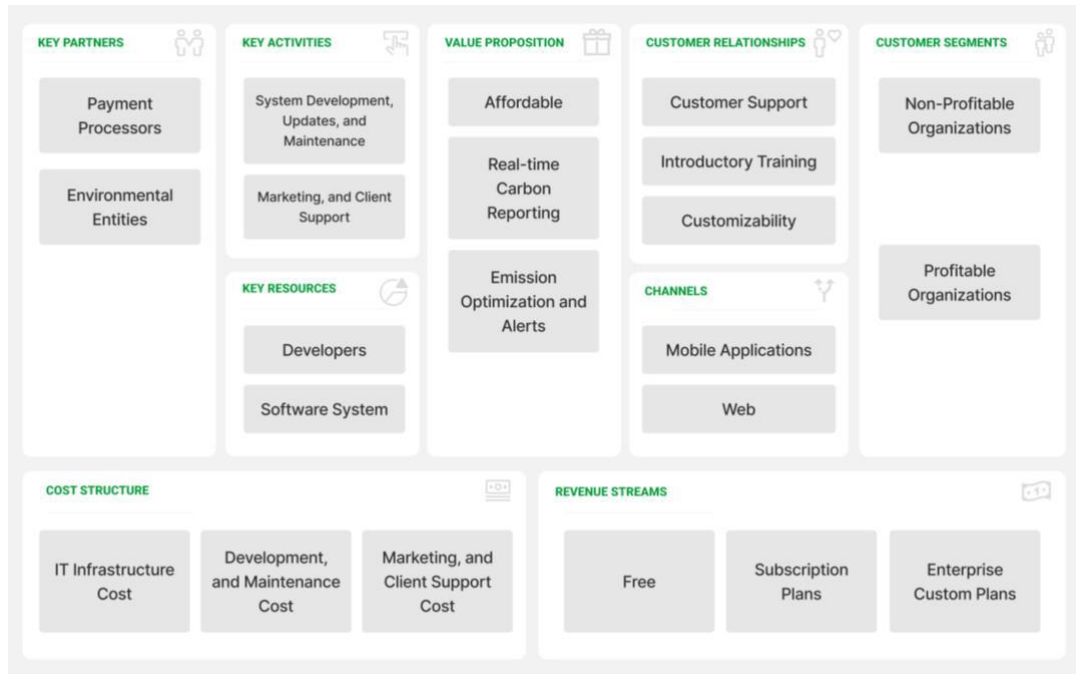


Figure 4.20: Business model canvas

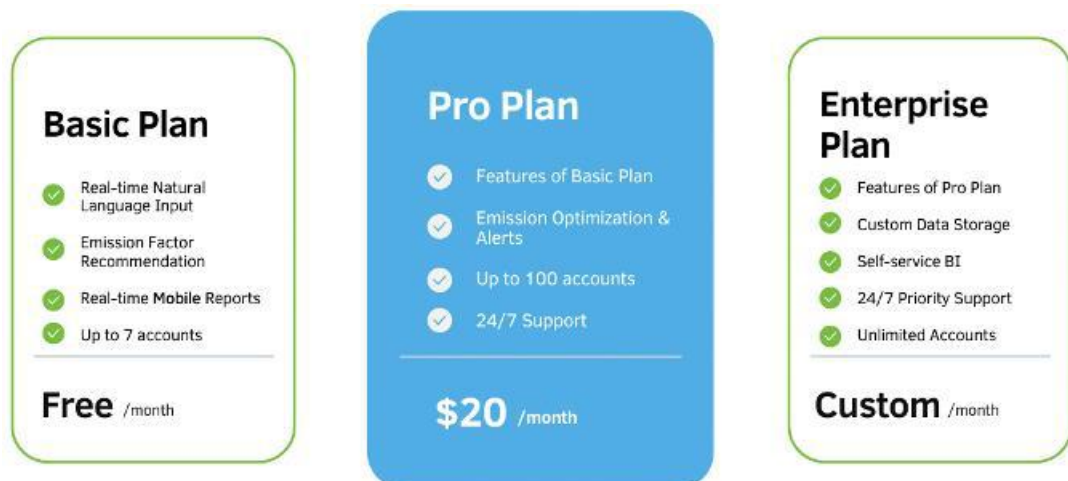


Figure 4.21: Pricing plan

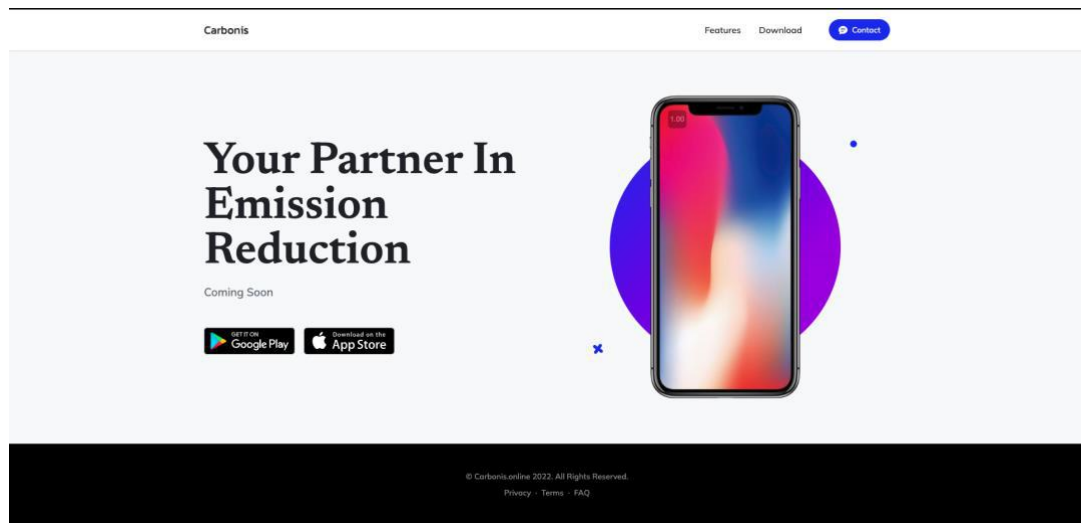


Figure 4.22: Product landing page



Figure 4.23: Product promotion pamphlet

4. Registering a professional domain address

Registered a web domain address (“www.carbonis.online”) for the landing and app deployment.

5. Creating a product landing page

Figure 4.22 shows the currently created product landing page deployed on the domain address using GitHub pages.

6. Designing a product pamphlet

As shown in Figure 4.23, designed a promotion pamphlet to promote the product locally.

Future commercialization tasks are as follows,

1. Search Engine Optimization (SEO) for the landing page
2. Social media promotion
3. Creating a demo video
4. Creating a commercial advertisement video
5. Distributing promotional pamphlets

5. RESULTS AND DISCUSSION

This chapter presents experiment results in the results section and the findings of those experiments in the research finding section. Finally, the discussion section summarizes the finding and the reasoning behind these findings.

5.1. Results

User satisfaction: Figure 5.1 and Table 5.1 lists the initial run results for 11 pre-trained and one custom-trained (word2vec-wiki-custom-defra-150) word vectors. It includes the maximum MAP, delta value for maximum MAP, best delta value range, MAP with no VSM (only embedding score), and storage usage by each word vector. Figure 5.2 and Table 5.2 shows a more refined maximum MAP and best delta value for the four best-performing word vectors on the second run. For the automated optimization, gradient ascend is unsuitable as there is no evaluable relationship between delta and MAP. However, other optimization algorithms from surrogate models, such as the gaussian process, tree-based search, and grid search, worked successfully. Among these, the Gaussian process provided the best possible findings within the shortest time or iterations. Figure 5.3 shows the results of such optimization. It turned out that the manual and automated optimization provided the same results.

Table 5.1: Summarized coarse MAP results for word vectors and delta values

Word Vector	Best MAP	Best MAP Delta	Best Delta Range	No VSM MAP	Size (MB)
conceptnet-numberbatch-17-06-300 [33]	0.60	0	0 – 0.95	0.11	2372
fasttext-wiki-news-subwords-300 [32]	0.69	0.25	0.25 – 0.3	0.17	1229
glove-twitter-25 [30]	0.63	0.05	0.05 - 0.15	0.13	157
glove-twitter-50 [30]	0.65	0.3	0.25 – 0.3	0.15	276
glove-twitter-100 [30]	0.67	0.25	0.25 – 0.3	0.14	515
glove-twitter-200 [30]	0.68	0.3	0.25 – 0.35	0.18	992
glove-wiki-gigaword-50 [30]	0.69	0.3	0.3 – 0.35	0.21	91
glove-wiki-gigaword-100 [30]	0.73	0.35	0.3 – 0.35	0.27	171

glove-wiki-gigaword-200 [30]	0.78	0.4	0.4 – 0.45	0.33	331
glove-wiki-gigaword-300 [30]	0.81	0.4	0.4 – 0.45	0.36	491
word2vec-google-news-300 [29], [35], [36]	0.77	0.4	0.4 – 0.45	0.31	3704
word2vec-wiki-custom-defra-150	0.70	0.25	0.2 – 0.25	0.29	419

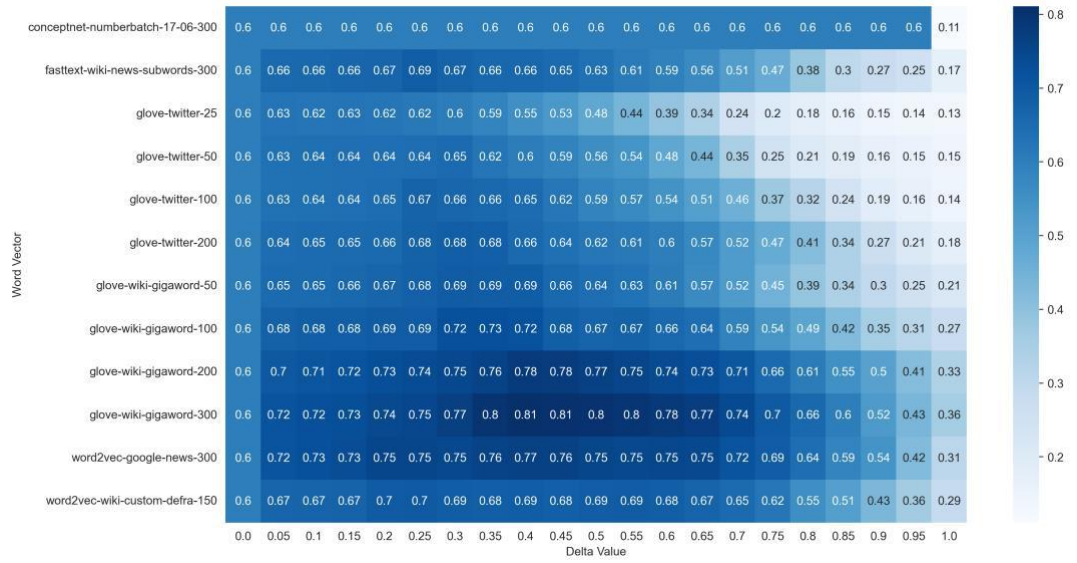


Figure 5.1: Coarse MAP results for word vectors and delta values

Table 5.2: Fine MAP results for word vectors and delta values for best word vectors

Word Vector	Best MAP	Best MAP Delta
glove-wiki-gigaword-100	0.73	0.35
glove-wiki-gigaword-200	0.78	0.37
glove-wiki-gigaword-300	0.81	0.41
word2vec-google-news-300	0.77	0.39

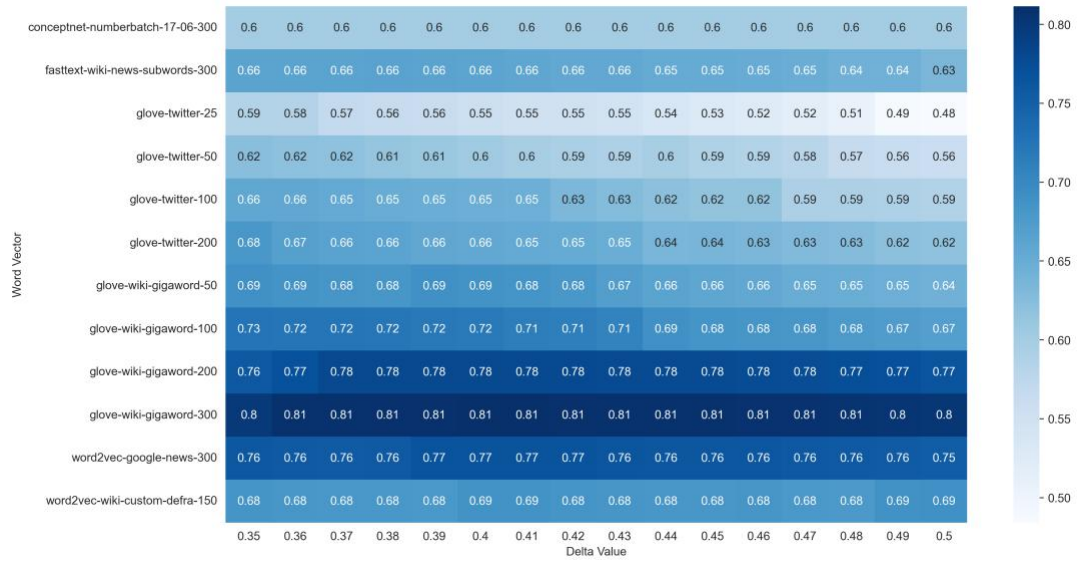


Figure 5.3: Fine MAP results for word vectors and delta values

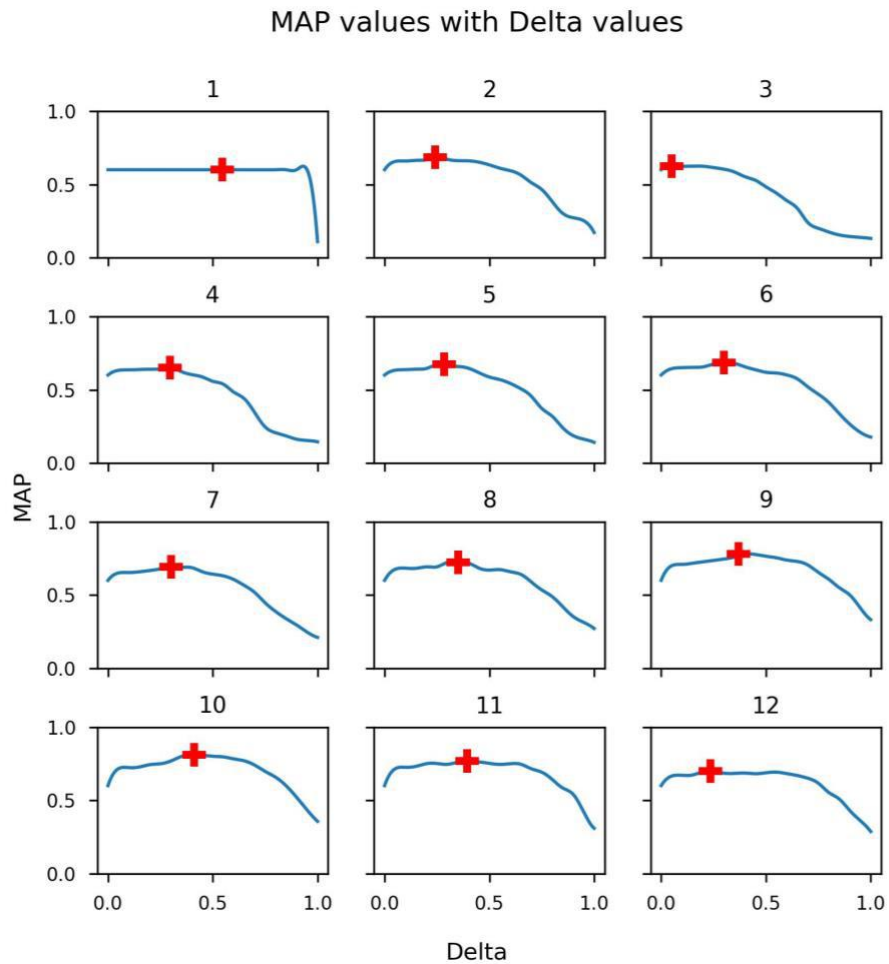


Figure 5.2: Optimization results visualization for user satisfaction using MAP

Average query speed: Wall time had noises caused by other system applications and a maximum error range of 10 milliseconds from CPU time. Therefore, observation involved only using CPU times. Figure 5.4 and Figure 5.5 illustrates average time results with different term and EF document counts. Figure 5.6 shows the average times of various word vectors and delta values. In addition, DEFRA and IPCC had average times of nearly 170 and 540 milliseconds, respectively. Table 5.3 shows the average times for EF datasets with the highest EF document and terms counts.

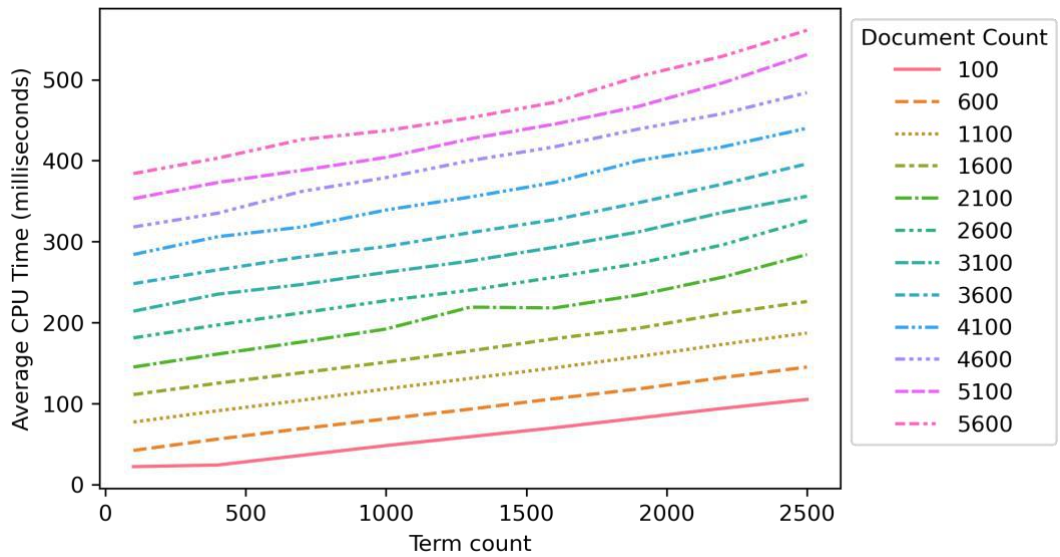


Figure 5.5: Average CPU time with term count

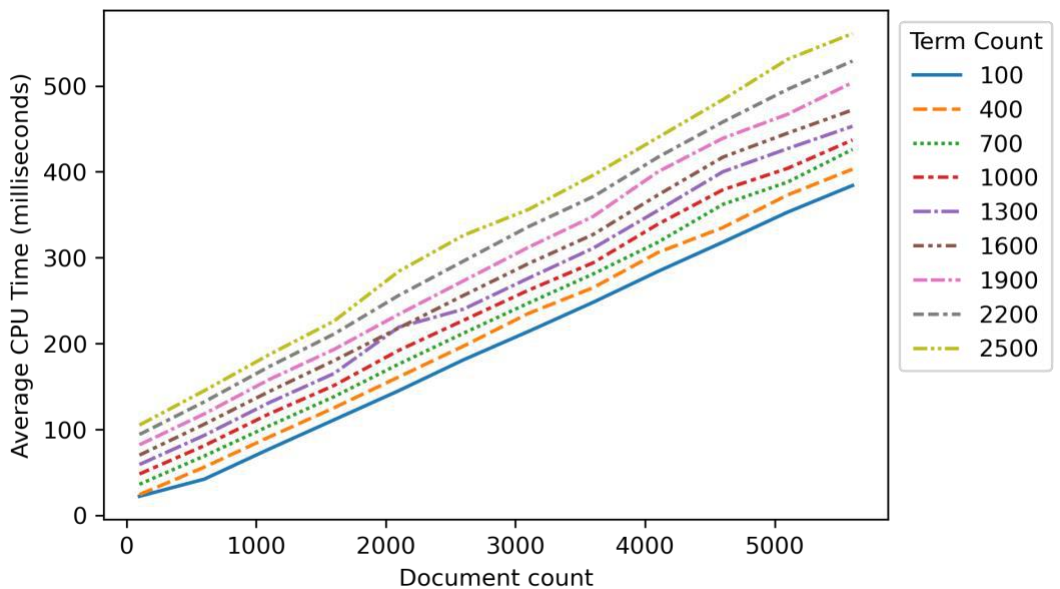


Figure 5.4: Average CPU time with document count

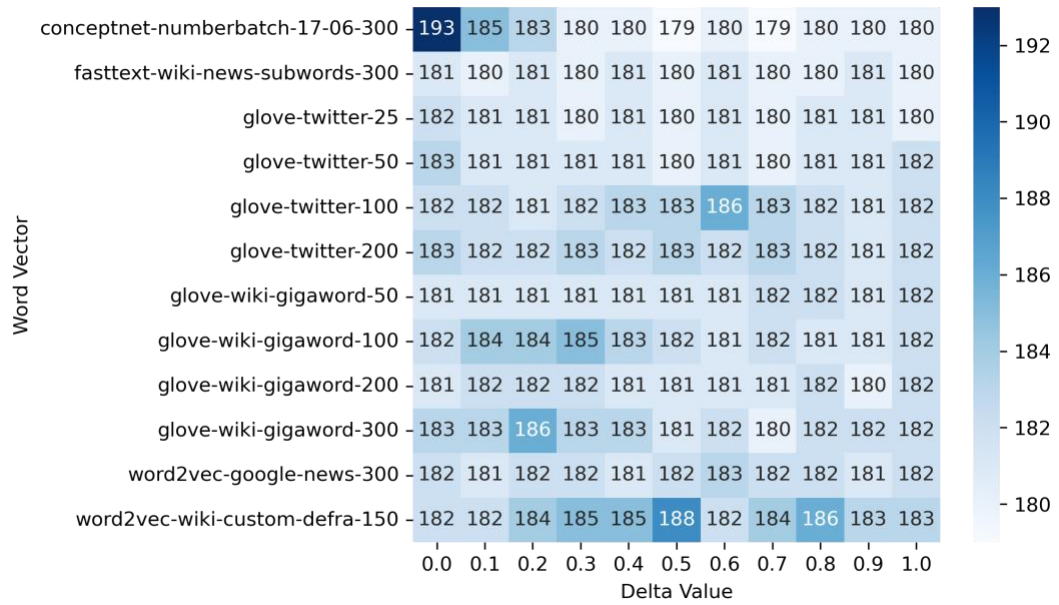


Figure 5.6: Average CPU time with word vector and delta values

Table 5.3: Average query time for vast EF datasets

EF Standard	EF Document Count	Term Count	Average Time (milliseconds)
DEFRA 2021	2435	365	181
IPCC 2006	6201	2508	536

EF scalability: Adoption of IPCC (1996 and 2006) took approximately 12 person-hours with no code modification except the PC1 part. However, PC1 is supposed to have some changes during the new EF adoption.

System resource utilization: Table 5.1 includes the storage usage of the word vectors, and Table 5.4 shows the memory and storage usage of TF-IDF matrices. TF-IDF matrices use the same amount of storage and memory.

Table 5.4: Memory and storage usage by TF-IDF matrices

EF Standards	Year	Matrix Size (document count, term count)	Memory or Storage Usage (MB)	Total Standard Usage (MB)
DEFRA	2014	1953, 332	5.2	50.4

	2015	2015, 361	5.8	
	2016	1989, 336	5.4	
	2017	2353, 346	6.5s	
	2018	2389, 353	6.8	
	2019	2390, 354	6.8	
	2020	2367, 357	6.8	
	2021	2435, 365	7.1	
IPCC	1996	6201, 2508	124.5	248.9
	2006	6201, 2508	124.5	

5.2. Research Findings

User satisfaction:

- **Effect of word vector and delta:** User satisfaction changes with word vector and delta values.
- **Best word vector:** glove-wiki-gigaword-300 outperformed others with a MAP of 0.81, nearly 30% improvement over plane VSM (0.60 MAP), and 127% improvement over plane word embedding (0.36 MAP). glove-wiki-gigaword (100, 200) and word2vec-google-news-300 also performed well.
- **Best delta value:** Even though the best delta range changes with word vector, most word vectors performed well between the 0.35 and 0.5 delta range, and glove-wiki-gigaword-300 gave the highest MAP at 0.41 delta value.
- **Improvement by combined rank approach:** With the combined rank approach, almost all word vectors improved MAP except conceptnet-numberbatch-17-06-300 (no improvement for any delta) over plane VSM or Embedding scoring.
- **Effect of word vector domain, size, dimension, and algorithms:** Wikipedia corpus and GloVe algorithm seem most relevant, and size and dimensions improve results.
- **Custom word vector improvement:** Custom-trained word vectors also showed an improved 0.70 MAP for 0.25 delta.
- **Best optimization method:** Gaussian process provided the best findings considering the lower iteration count taken to converge on the maximum MAP values.

Average query speed:

- **Effect of EF document and term counts:** Average time increases with EF document and terms counts. However, even the vast EF datasets of DEFRA and IPCC only had sub-second average times. Therefore, the average speed can be considered acceptable for current EF datasets. Can use heuristic optimization techniques for future changes.
- **Effect of word vector and delta:** There was no noticeable impact on average speed by word vector or delta value. Can neglect slight deviations in favor of measuring error and other factors.

EF scalability:

- **EF scalability:** Current EF retrieval system adopts new EF standards and datasets with minimal effort and code changes.

System resource utilization:

- **The best scalable word vector:** glove-wiki-gigaword-300, wins with low resource utilization considering its high user satisfaction. Furthermore, its storage usage is less than 1/7th of word2vec-google-news-30. Therefore, glove-wiki-gigaword-300 can be considered the best scalable word vector among these word vectors.
- **Need for TF-IDF resource utilization:** Even though TF-IDF matrices' resource utilizations are acceptable, optimization and compression techniques can improve future scalability, especially with IPCC's TF-IDF matrices. However, most EF datasets are not extensive as IPCC.

5.3. Discussion

The EF retrieval system's combined ranking approach showed improved user satisfaction over plane VSM and word embedding. As explained in the word embedding preliminary, corpus domain, corpus size, dimension, and algorithms affect user satisfaction, and Wikipedia corpus, GloVe, or word2vec algorithm, decent corpus size and dimension provides better user satisfaction. Query speed is acceptable and

decreases with EF document and term counts. EF retrieval system scales for new EF standards, datasets, and user space. The gaussian process can successfully automate finding the best word vectors and delta parameter values. The AWS personalize trained models could further improve the rankings. However, experiments and improvements are needed, e.g., system resource utilization improvements.

6. CONCLUSIONS

This research proves the hypothesis that the combined ranking approach will improve usability and be scalable for EF selection tasks compared to plane VSM or word embedding. This work provides a novel scalable EF retrieval system with a combined ranking approach for better usability for corporate uses, evaluation metrics for such systems, and evaluation results and findings. It is possible to improve the usability and scalability of the emission calculation interface with this EF retrieval system. Can expect to save user time and effort during emission calculation. An improvement of 29.95% and 127% was observed over VSM and embedding rankings, respectively, by the combined rank approach with glove-wiki-gigaword-300 for a 0.41 linear combination parameter (delta) value. All queries took a sub-second delay for DEFRA (2014 to 2021) and IPCC (1996 and 2006). Similar surrogate optimization methods could automate ranking evaluation. However, further experimentations and improvements, such as performance and memory optimizations, are needed.

Future tasks include scaling other EF standards and experimenting with this retrieval system. Additionally, the plan includes applying heuristic optimizations to improve performance and scalability. Finally, the author hopes that this work is contributed to and further improved by the research community.

REFERENCES

- [1] M. Roelfsema *et al.*, “Taking stock of national climate policies to evaluate implementation of the Paris Agreement,” *Nature Communications* 2020 11:1, vol. 11, no. 1, pp. 1–12, Apr. 2020, doi: 10.1038/s41467-020-15414-6.
- [2] C. F. Schleussner *et al.*, “Science and policy characteristics of the Paris Agreement temperature goal,” *Nature Climate Change* 2016 6:9, vol. 6, no. 9, pp. 827–835, Jul. 2016, doi: 10.1038/nclimate3096.
- [3] “SRI LANKA UPDATED NATIONALLY DETERMINED CONTRIBUTIONS”.
- [4] B. Doda, C. Gennaioli, A. Gouldson, D. Grover, and R. Sullivan, “Are Corporate Carbon Management Practices Reducing Corporate Carbon Emissions?,” *Corp Soc Responsib Environ Manag*, vol. 23, no. 5, pp. 257–270, Sep. 2016, doi: 10.1002/CSR.1369.
- [5] E. P. Olaguer, “Emission Inventories,” *Atmospheric Impacts of the Oil and Gas Industry*, pp. 67–77, Jan. 2017, doi: 10.1016/B978-0-12-801883-5.00007-3.
- [6] N. R. A. Rahman, S. Z. A. Rasid, and R. Basiruddin, “Exploring the Relationship between Carbon Performance, Carbon Reporting and Firm Performance: A Conceptual Paper,” *Procedia Soc Behav Sci*, vol. 164, pp. 118–125, Dec. 2014, doi: 10.1016/J.SBSPRO.2014.11.059.
- [7] S. J. Arceivala, *Opportunities in Control of Carbon Emissions and Accumulation*. McGraw-Hill Education, 2014. Accessed: Jul. 17, 2022. [Online]. Available: <https://www.accessengineeringlibrary.com/content/book/9781259063732/chapter3>
- [8] T. Gao, Q. Liu, and J. Wang, “A comparative study of carbon footprint and assessment standards,” *International Journal of Low-Carbon Technologies*, vol. 9, no. 3, pp. 237–243, Sep. 2014, doi: 10.1093/IJLCT/CTT041.
- [9] J. Downie and W. Stubbs, “Corporate Carbon Strategies and Greenhouse Gas Emission Assessments: The Implications of Scope 3 Emission Factor Selection,” *Bus Strategy Environ*, vol. 21, no. 6, pp. 412–422, Sep. 2012, doi: 10.1002/BSE.1734.

- [10] C. C. Spork, A. Chavez, X. G. Durany, M. K. Patel, and G. V. Méndez, "Increasing Precision in Greenhouse Gas Accounting Using Real-Time Emission Factors," *J Ind Ecol*, vol. 19, no. 3, pp. 380–390, Jun. 2015, doi: 10.1111/JIEC.12193.
- [11] J. Frijns, "Towards a common carbon footprint assessment methodology for the water sector," *Water and Environment Journal*, vol. 26, no. 1, pp. 63–69, Mar. 2012, doi: 10.1111/J.1747-6593.2011.00264.X.
- [12] V. Franco, M. Kousoulidou, M. Muntean, L. Ntziachristos, S. Hausberger, and P. Dilara, "Road vehicle emission factors development: A review," *Atmos Environ*, vol. 70, pp. 84–97, May 2013, doi: 10.1016/J.ATMOSENV.2013.01.006.
- [13] N. P. Cheremisinoff, "Pollution Management and Responsible Care," *Waste*, pp. 487–502, Jan. 2011, doi: 10.1016/B978-0-12-381475-3.10031-2.
- [14] G. Bekaroo, D. Roopowa, and C. Bokhoree, "Mobile-Based Carbon Footprint Calculation: Insights from a Usability Study," *2nd International Conference on Next Generation Computing Applications 2019, NextComp 2019 - Proceedings*, Sep. 2019, doi: 10.1109/NEXTCOMP.2019.8883622.
- [15] A. C. Peres Vieira, E. M. F. da Silva, and V. V. V. Aguiar Odakura, "Development of a Web Application for Individual Carbon Footprint Calculation," *Proceedings - 2021 47th Latin American Computing Conference, CLEI 2021*, 2021, doi: 10.1109/CLEI53233.2021.9640099.
- [16] D. Andersson, "A novel approach to calculate individuals' carbon footprints using financial transaction data – App development and design," *J Clean Prod*, vol. 256, p. 120396, May 2020, doi: 10.1016/J.JCLEPRO.2020.120396.
- [17] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports," *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pp. 127–137, Dec. 2016, doi: 10.1109/ISSRE.2016.33.
- [18] J. Mulrow, K. Machaj, J. Deanes, and S. Derrible, "The state of carbon footprint calculators: An evaluation of calculator design and user interaction features," *Sustain Prod Consum*, vol. 18, pp. 33–40, Apr. 2019, doi: 10.1016/J.SPC.2018.12.001.

- [19] D. Hu *et al.*, “Recommending Similar Bug Reports: A Novel Approach Using Document Embedding Model,” *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2018-December, pp. 725–726, Jul. 2018, doi: 10.1109/APSEC.2018.00108.
- [20] Y. Wang *et al.*, “A comparison of word embeddings for the biomedical natural language processing,” *J Biomed Inform*, vol. 87, pp. 12–20, Nov. 2018, doi: 10.1016/J.JBI.2018.09.008.
- [21] Defra, “Environmental Reporting Guidelines,” 2019, Accessed: Jan. 23, 2022. [Online]. Available: www.nationalarchives.gov.uk/doc/open-government-licence/
- [22] M. Bhavadharani, M. P. Ramkumar, and S. G. S. R. Emil, “Performance analysis of ranking models in information retrieval,” *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, pp. 1207–1211, Apr. 2019, doi: 10.1109/ICOEI.2019.8862785.
- [23] L. Xiaoli, Y. Xiaokai, and L. Kan, “An improved model of document retrieval efficiency based on information theory,” *J Phys Conf Ser*, vol. 1848, no. 1, p. 012094, Apr. 2021, doi: 10.1088/1742-6596/1848/1/012094.
- [24] R. Ali *et al.*, “Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents Text Mining,” *Article in International Journal of Computer Applications*, vol. 181, no. 1, pp. 975–8887, 2018, doi: 10.5120/ijca2018917395.
- [25] M. Farouk, “Measuring text similarity based on structure and word embedding,” *Cogn Syst Res*, vol. 63, pp. 1–10, Oct. 2020, doi: 10.1016/J.COGSYS.2020.04.002.
- [26] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. J. Kuo, “Evaluating word embedding models: methods and experimental results,” *APSIPA Trans Signal Inf Process*, vol. 8, pp. 1–14, 2019, doi: 10.1017/ATSIP.2019.12.
- [27] S. Lai, K. Liu, S. He, and J. Zhao, “How to generate a good word embedding,” *IEEE Intell Syst*, vol. 31, no. 6, pp. 5–14, Nov. 2016, doi: 10.1109/MIS.2016.45.
- [28] Y. Y. Lee, H. Ke, T. Y. Yen, H. H. Huang, and H. H. Chen, “Combining and learning word embedding with WordNet for semantic relatedness and similarity

- measurement,” *J Assoc Inf Sci Technol*, vol. 71, no. 6, pp. 657–670, Jun. 2020, doi: 10.1002/ASI.24289.
- [29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, Jan. 2013, doi: 10.48550/arxiv.1301.3781.
 - [30] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” pp. 1532–1543, Accessed: Jul. 22, 2022. [Online]. Available: <http://nlp>.
 - [31] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of Tricks for Efficient Text Classification.” [Online]. Available: <https://github.com/facebookresearch/>
 - [32] T. Mikolov, E. Grave, P. Bojanowski, C. Puhersch, and A. Joulin, “Advances in Pre-Training Distributed Word Representations,” *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pp. 52–55, Dec. 2017, doi: 10.48550/arxiv.1712.09405.
 - [33] R. Speer, J. Chin, and C. Havasi, “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Dec. 2016, doi: 10.48550/arxiv.1612.03975.
 - [34] M. Sanderson and J. Zobel, “Information retrieval system evaluation: Effort, sensitivity, and reliability,” *SIGIR 2005 - Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 162–169, 2005, doi: 10.1145/1076034.1076064.
 - [35] T. Mikolov, W.-T. Yih, and G. Zweig, “Linguistic Regularities in Continuous Space Word Representations”, Accessed: Jul. 31, 2022. [Online]. Available: <http://research.microsoft.com/en->
 - [36] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *Adv Neural Inf Process Syst*, Oct. 2013, doi: 10.48550/arxiv.1310.4546.

APPENDICES

Appendix A. Emission Factor Retrieval Tests

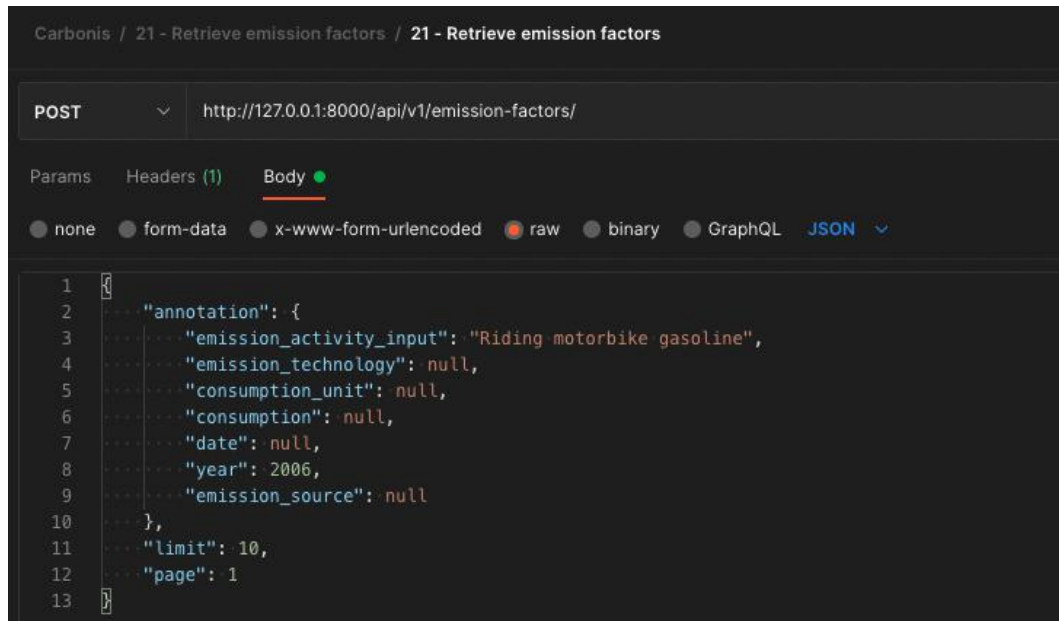


Figure 0.1: EF retrieval API endpoint's sample request (Postman)

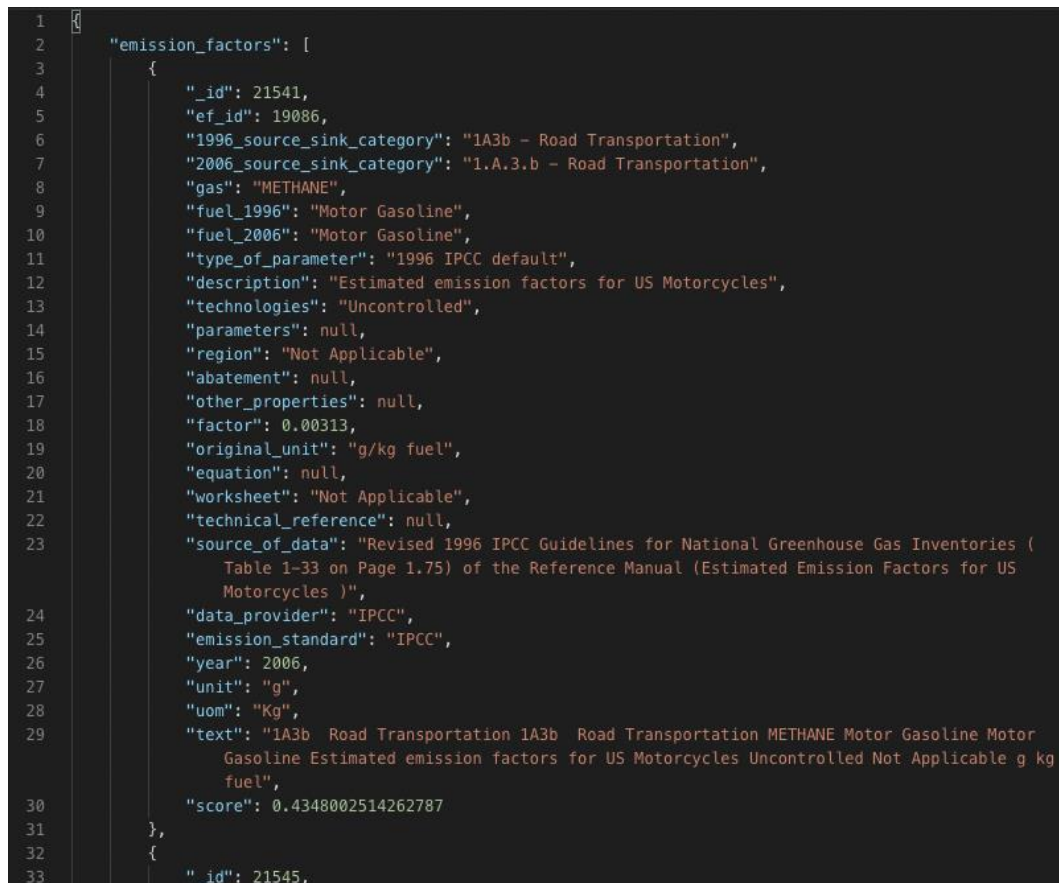


Figure 0.2: EF retrieval API endpoint's sample response (Postman)

Appendix B. Emission Calculation Tests

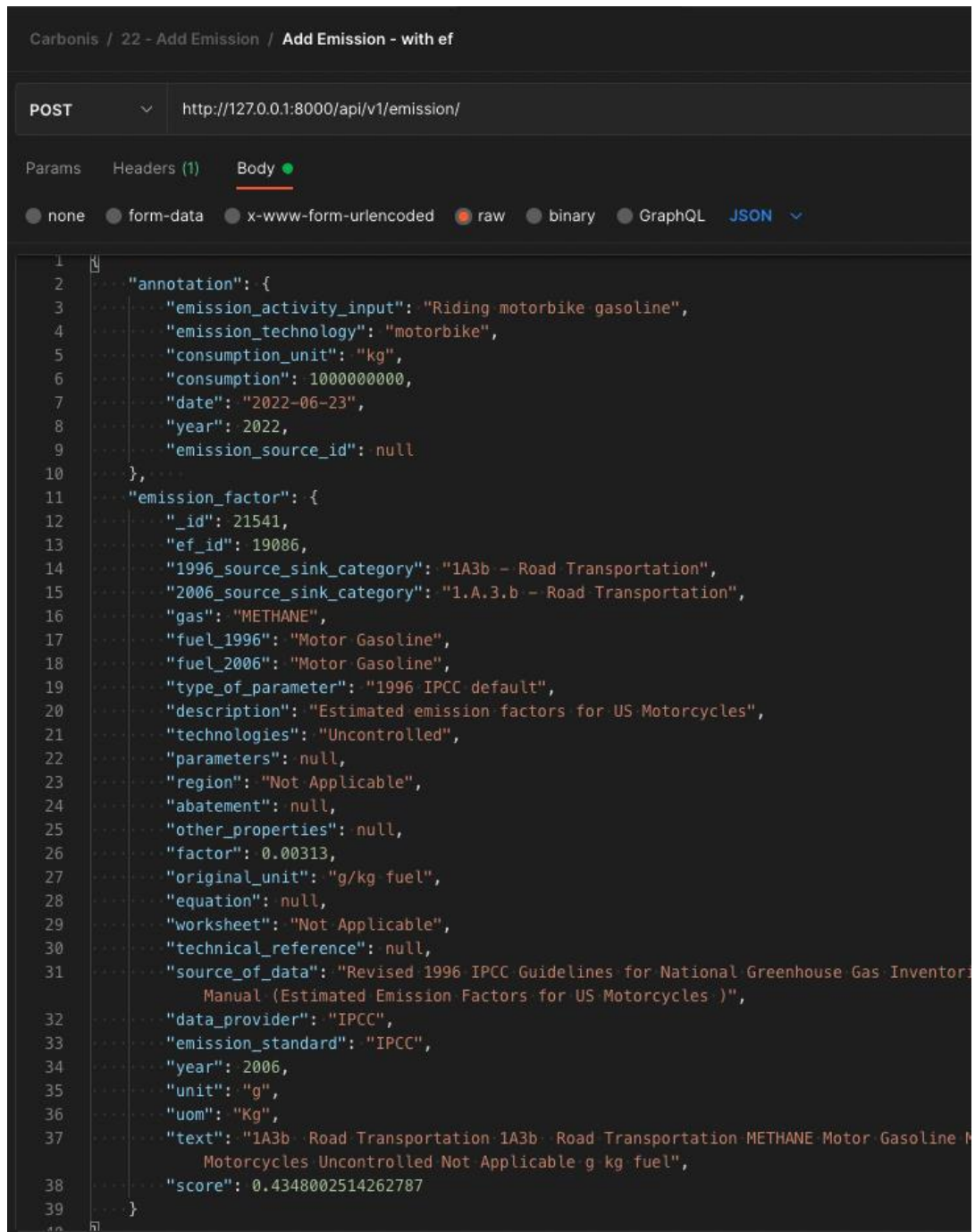


Figure 0.3: Emission calculation API endpoint's sample request (Postman)

```

1  {
2    "id": 17,
3    "user": {
4      "id": 2,
5      "username": "emp1",
6      "first_name": "",
7      "last_name": "",
8      "email": ""
9    },
10   "division": {
11     "id": 3,
12     "name": "div1",
13     "location": null,
14     "level": 1,
15     "emission_cap": null,
16     "created_date": "2022-08-24T16:32:08.123499Z",
17     "modified_date": "2022-08-24T16:32:08.123600Z",
18     "top_div": null,
19     "parent_div": null,
20     "organization": 1
21   },
22   "emission_source": null,
23   "technology": "motorbike",
24   "consumption_unit": "kg",
25   "consumption": "100000000.00",
26   "emission_factor": "0.00",
27   "emission_factor_doc_id": 21541,
28   "emission": "3130.00",
29   "date": "2022-06-23",
30   "division_level": 1,
31   "created_date": "2022-08-28T13:30:15.706933Z",
32   "modified_date": "2022-08-28T13:30:15.706978Z"
33 }

```

Figure 0.4: Emission calculation API endpoint's sample response (Postman)

Appendix C. Emission Flow UI Prototypes

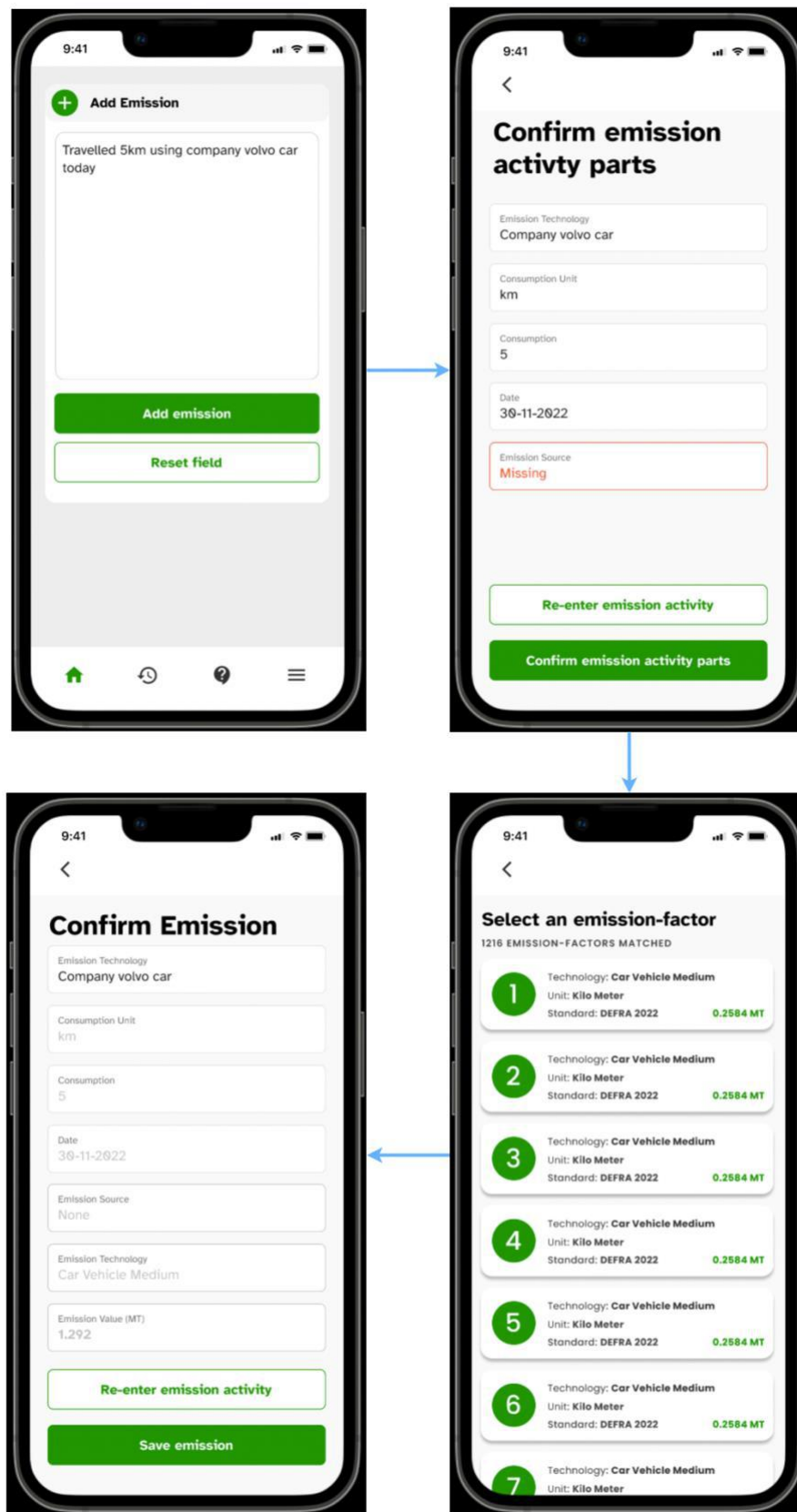


Figure 0.5: Frontend UI prototype for emission calculation and EF retrieval

Appendix D. Sample MAP evaluation dataset

Table 0.1: Three Sample MAP evaluation queries and relevant EF documents

Sample Query	Expected Relevant EF Documents							
	Scope	Level 1	Level 2	Level 3	Level 4	Column Text	UOM	Factor (Kg CO2-eq)
Cooked using compressed natural gas fuel	Scope 1	Fuels	Gaseous fuels	CNG		Volume	litres	0.44423
	Scope 3	WTT-fuels	WTT-liquid fuels	Gas Oil		Volume	litres	0.63253
	Scope 3	WTT-fuels	WTT-gaseous fuels	CNG		Energy - Gross CV	kWh (Gross CV)	0.03912
Flown using jet fuel	Scope 1	Fuels	Liquid fuels	Aviation turbine fuel		Volume	litres	2.54514
	Scope 3	WTT-fuels	WTT-liquid fuels	Aviation Spirit		Energy - Net CV	kWh (Net CV)	0.06552
Using petrol gas lamp	Scope 1	Fuels	Liquid fuels	Burning oil		Energy - Net CV	kWh (Net CV)	0.25975
	Scope 3	WTT-fuels	WTT-liquid fuels	Gas Oil		Volume	litres	0.63253
	Scope 1	Fuels	Gaseous fuels	Other petroleum gas		Tonnes	tonnes	2578.25
	Scope 1	Fuels	Liquid fuels	Petrol (100% mineral petrol)		Volume	litres	2.33969
	Scope 1	Fuels	Liquid fuels	Petrol (100% mineral petrol)		Energy - Gross CV	kWh (Gross CV)	0.24158