National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Priscila Cristina da Silva de Oliveira
x23157003

School of Computing
National College of Ireland

Supervisor: Barry Haycock

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Priscila Cristina da Silva de Oliveira
Student ID:	x23157003
Programme:	MSc in Data Analytics **Year:** 2024
Module:	Research Project
Lecturer:	Barry Haycock
Submission Due Date:	16.09.2024
Project Title:	From LDA to BERTopic: Evaluating Topic Modelling Methods for Aviation Safety Reports in Brazilian Portuguese
Word Count:	3149 **Page Count:** 17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Priscila C. D. Oliveira

.....

Date: 16.09.2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Priscila Cristina da Silva de Oliveira
x23157003

1 System Specifications

This section details the hardware and operating system used in this study. The hardware specifications are not minimum requirements. However, one should be aware that different specifications could result on different performance.

1.1 Hardware Specifications

Processor	13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz
RAM	64.0 GB
GPU	NVIDIA GeForce RTX 4070 Laptop GPU

1.2 Software Environment

Operating System	Windows 11 Pro, 64-bit
Integrated Development Environment	PyCharm 2024.1.1
Jupyter Notebook	6.5.4
Python Version	3.11.7

1.2.1 Key libraries

Library / Package	Version
adjustText	1.2.0
beautifulsoup4	4.12.2
bertopic	0.16.3
gensim	4.3.0
matplotlib	3.8.0
nltk	3.8.1
numpy	1.26.4
pandas	2.1.4
pyLDAvis	3.4.1
PyMuPDF	1.24.7
seaborn	0.12.2
scikit-learn	1.2.2
scipy	1.12.0
resquests	2.31.0

tqdm	4.65.0
umap-learn	0.5.6
wordcloud	1.9.3

2 Dataset Characteristics

The aviation safety reports used for the research project are publicly available on the website https://sistema.cenipa.fab.mil.br/cenipa/paginas/relatorios/relatorios.php in the PDF format. The documents are distributed over 31 tabs and most of the files are in Portuguese. Figure 1 shows a snippet of the webpage. The flags represent the language in which the report is available.

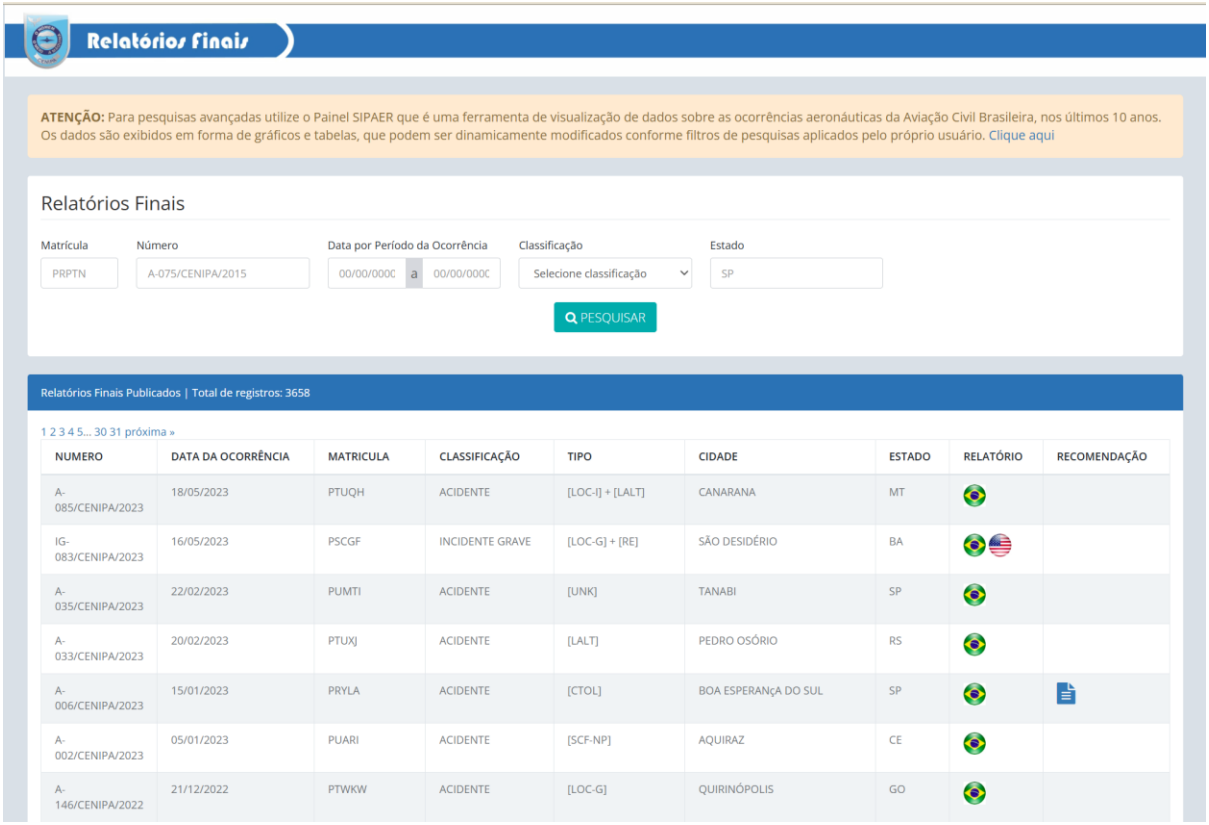

Figure 1 Reports in CENIPA website

Some documents are available in English and Spanish. It was also observed that some of the files, especially those reporting older events, are scanned documents which are not machine-readable. In addition, during the download phase, it was noted that a small number of documents cannot be accessed although the instance is shown in the reports table.

The download of the files was performed using web scraping leveraging BeautifulSoup4. The first version of the code used did not consider the different existing tabs and, for this reason, only the documents in the first tab were downloaded. The steps followed are described in Algorithm 1. The code described in the algorithm is available in the file named download_pdf_version_1.ipynb.

Algorithm 1 Download PDF files from CENIPA website – version 1

```
FUNCTION main():

    SET URL to "https://sistema.cenipa.fab.mil.br/cenipa/paginas/relatorios/relatorios.php"

    response = GET request to URL

    IF response status is not successful THEN
        PRINT error message
        RETURN false

    page = Parse HTML content of response

    table = Find element in page with ID "lista"

    links = Extract all <a> elements from table

    FOR EACH link in links:

        href = Get "href" attribute of link

        IF "pdf" is not in href THEN
            CONTINUE to next iteration

        filename = Remove path from href, keeping only the file name

        pdfUrl = Concatenate "https://sistema.cenipa.fab.mil.br/cenipa/paginas/relatorios/"
        and href

        pdfContent = GET request to pdfUrl

        Write pdfContent to file named filename

        PRINT "Getting {filename}..."

    RETURN true

CALL main()
```

Given that not all the documents available are machine-readable and that this project focuses on the reports written in the Portuguese language, the initial code was modified to:

- Classify the reports between “Portuguese” and “Other Languages”
- Subclassify the reports between “Scanned” and “Searchable”
- Access the 31 existing tabs to download all the files available.

The final code for the web scrapping process verifies whether the file exists before proceeding with the download. If so, it skips to the next file. It also informs in cases when it fails to download the document.

In order to verify whether a certain file is machine-readable, the function *is_searchable()* was defined. The *main()* function defines the folder and subfolder names. It verifies if the main folders exist, before creating them.

Next, it iterates over the 31 existing tabs using *BeautifulSoup* to parse the HTML content. If the PDF link is valid, it categorises the file and save or skip the document. Algorithm 2 describes the steps used to download the files used in the study and the code can be found in the file download_pdf_final_version.ipynb.

Algorithm 2 Download PDF files from CENIPA website – final version

```
FUNCTION is_searchable(pdf_path):

    TRY:
        Open PDF at pdf_path
        IF PDF has no pages THEN
            RETURN false
        Get text from first page
        RETURN true if text is not empty, false otherwise
    CATCH IndexError:
        PRINT error message
        RETURN false

FUNCTION main():
    SET portuguese_folder to 'PortugueseReports'
    SET other_languages_folder to 'Other_Languages'
    SET subfolders to ['Scanned', 'Searchable']

    CREATE folders and subfolders if they don't exist

    FOR page_number FROM 1 TO 31:
        SET URL to
        f"https://sistema.cenipa.fab.mil.br/cenipa/paginas/relatorios/relatorios?&&?&pag={p
        age_number}"

    Send GET request to URL
    IF request failed THEN
    PRINT error message
    RETURN false

    Parse HTML content of response
    Find table with id "lista"
    Extract all links from table

    FOR EACH link in links:
        IF link doesn't contain 'pdf' THEN
            CONTINUE to next link

        SET pdf_url to full URL of PDF
        Download PDF file

        IF download successful THEN
            Save PDF to temporary file

            IF link title is "Relatório Final em Português" THEN
                SET language_folder to portuguese_folder
            ELSE
                SET language_folder to other_languages_folder

            IF is_searchable(temporary_file) THEN
```

```
                SET save_subfolder to 'Searchable'
            ELSE
                SET save_subfolder to 'Scanned'

            SET save_path to language_folder/save_subfolder/pdf_filename

            IF save_path doesn't exist THEN
                Move temporary file to save_path
                PRINT success message
            ELSE
                Delete temporary file
                PRINT file already exists message

        ELSE
            PRINT download failure message

    RETURN true

CALL main()
```

3 Extracting Metadata – Types of Occurrences

The types of occurrences were manually extracted from CENIPA's website. The values were copied and pasted on Microsoft Excel. The unique values are shown in Table 1. The code ALTL was identifies among the types of occurrences. However, by analysing the report classified in the category, it became clear that it was a typing error. The occurrence should have been classified as LALT.

Table 1: Types of Occurrences

Type	Description
ADRM	aerodrome
AMAN	abrupt maneuvre
ARC	abnormal runway contact
ATM/CNS	atm/cns
BIRD	bird
CFIT	controlled flight into/toward terrain
CTOL	collision with obstacle(s) during take-off and landing
EXTL	external load related occurrences
F-NI	fire/smoke (non-impact)
F-POST	fire/smoke (post-impact)
FUEL	fuel related
GCOL	ground collision
GTOW	glider towing related events
ICE	icing
LALT	low altitude operations
LOC-G	loss of control – ground

LOC-I	loss of control – inflight
LOLI	loss of lifting conditions en-route
MAC	airprox/tcas alert/loss of separation/near midair collisions/midair collisions
MED	medical
OTHR	other
RAMP	ground handling
RE	runway excurision
RI	runway incursion
SCF-NP	system/component failure or malfunction (non-powerplant)
SCF-PP	system/component failure or malfunction (powerplant)
TURB	turbulence encounter
UIMC	unintended flight in imc
UNK	unknown or undetermined
USOS	undershoot/overshoot
WILD	collision with animals
WSTRW	wind shear or thunderstorm

4 LDA

Algorithm 3 describes the steps taken for the LDA model. The respective code is available in the file named LDA_final_version.ipynb.

Algorithm 3 Topic Modelling with LDA

```
SET seed to 23157003
INITIALIZE random number generators with seed

DOWNLOAD NLTK resources

SET pdf_folder_path to location of PDF files

DEFINE extra_stop_words
COMBINE extra_stop_words with Portuguese stopwords from NLTK

FUNCTION extract_text_from_pdf(pdf_path):
        INITIALIZE empty text string
        TRY:
                OPEN pdf_file
                FOR EACH page in pdf_file:
                        ADD page text to text string
        CATCH any exceptions:
                PRINT error message
        RETURN text string

FUNCTION preprocess(text):
        CONVERT text to lowercase
        REMOVE numbers from text
        REMOVE punctuation from text
        TOKENIZE text
        RETURN list of tokens not in stop_words

FUNCTION get_top_ngrams(processed_texts, n, top_k):
```

```
    CREATE empty list for n-grams
    FOR EACH text in processed_texts:
        GENERATE n-grams from text
        ADD n-grams to list
    COUNT frequency of n-grams
    RETURN top_k most common n-grams

FUNCTION compute_coherence_and_perplexity(dictionary, corpus, texts, start, limit, step):
    INITIALIZE empty lists for coherence_values, perplexity_values, and model_list
    FOR num_topics FROM start TO limit STEP step:
        CREATE LdaModel with num_topics
        COMPUTE coherence score
        COMPUTE perplexity
        ADD scores and model to respective lists
    RETURN model_list, coherence_values, perplexity_values

FUNCTION evaluate_lda_model(corpus, dictionary, processed_texts, start, limit, step):
    CALL compute_coherence_and_perplexity
    NORMALIZE coherence and perplexity scores
    COMPUTE composite scores
    FIND best model based on highest composite score
    PRINT best model details
    PLOT evaluation metrics
    RETURN best_model, best_num_topics, best_coherence, best_perplexity

FUNCTION plot_evaluation_metrics(start, limit, step, coherence_values, perplexity_values,
composite_scores):
    CREATE plot with three y-axes
    PLOT coherence scores
    PLOT perplexity scores
    PLOT composite scores
    ADD labels and legend
    DISPLAY plot

FUNCTION visualize_topics(lda_model, dictionary, corpus, num_topics, num_words):
    GENERATE word clouds for all topics
    GENERATE bar charts for all topics
    CREATE intertopic distance map using t-SNE
    DISPLAY visualizations

MAIN PROCESS:
    INITIALIZE empty lists for processed_texts and raw_texts

    FOR EACH PDF file in pdf_folder_path:
    text = extract_text_from_pdf(file)
    ADD text to raw_texts
    processed_text = preprocess(text)
    ADD processed_text to processed_texts

    FOR n FROM 2 TO 5:
        COMPUTE and PRINT top 10 n-grams

    CREATE dictionary from processed_texts
    CREATE corpus from processed_texts and dictionary
```

best_model, best_num_topics, best_coherence, best_perplexity = evaluate_lda_model(corpus, dictionary, processed_texts)

CALL visualize_topics with best_model

PRINT best model's topics

SAVE best model to file

GENERATE and DISPLAY pyLDAvis visualization

5 LDA with Stemming

Algorithm 4 describes the pre-processing function for the LDA model with stemming. The remainder of the steps is equivalent to Algorithm 3. The respective code is available in LDA_stemming_final_version.ipynb.

Algorithm 4 Function preprocess(text) with Stemming

```
FUNCTION preprocess(text):
    CONVERT text to lowercase

    REMOVE numbers from text

    REMOVE punctuation characters from text

    TOKENIZE the lowercase text into individual words

    INITIALIZE empty list for processed_tokens

    FOR EACH word in tokens:
        IF word consists only of alphabetic characters AND word is not in stop_words:
            stem = APPLY stemming to word
            ADD stem to processed_tokens

    RETURN processed_tokens
```

6 LDA with Translation

Algorithm 5 details the cross-language model in which the reports were translated to English before performing LDA. The respective code is available in the file translation_lda_final_version.ipynb.

The functions *ensure_json_serializable*, *clean_vis_data_in_place* and *remove_complex_numbers* were created to solve the error TypeError: Object of type complex is not JSON serializable that was that was observed when trying to plot pyLDAvis.

Algorithm 5 Topic Modelling with LDA and Translation

```
SET seed to 23157003
```

```
INITIALIZE random number generators with seed

DOWNLOAD NLTK resources (punkt, stopwords, wordnet, omw-1.4)

DEFINE custom_stopwords
COMBINE custom_stopwords with English stopwords from NLTK

SET pdf_directory to location of PDF files
SET translated_texts_file to "translated_texts.pkl"

FUNCTION extract_text_from_pdfs(pdf_directory):
        INITIALIZE empty list text_data
        FOR EACH PDF file in pdf_directory:
                TRY:
                        OPEN PDF file
                        FOR EACH page in PDF:
                                EXTRACT text from page
                                ADD text to text_data
                        CATCH any exceptions:
                                PRINT error message
        RETURN text_data

FUNCTION translate_text(text_data, src_lang="pt", tgt_lang="en"):
        LOAD pre-trained translation model and tokenizer
        INITIALIZE empty list translated_texts
        FOR EACH text in text_data:
                TRY:
                        TOKENIZE and ENCODE text
                        GENERATE translation
                        DECODE translated tokens
                        ADD translated text to translated_texts
                CATCH any exceptions:
                        PRINT error message
                        ADD empty string to translated_texts
        RETURN translated_texts

FUNCTION preprocess_text(text):
        CONVERT text to lowercase
        REMOVE numbers
        REMOVE punctuation
        TOKENIZE text
        REMOVE stopwords and short words
        RETURN processed tokens

FUNCTION process_texts(pickle_file_path):
        LOAD texts from pickle file
        INITIALIZE empty list processed_texts
        FOR EACH text in texts:
                processed_tokens = preprocess_text(text)
                ADD processed_tokens to processed_texts
        RETURN processed_texts

FUNCTION compute_coherence_and_perplexity(dictionary, corpus, texts, start, limit, step):
        INITIALIZE empty lists for coherence_values, perplexity_values, and model_list
        FOR num_topics FROM start TO limit STEP step:
```

```
            CREATE LdaModel with num_topics
            COMPUTE coherence score
            COMPUTE perplexity
            ADD scores and model to respective lists
    RETURN model_list, coherence_values, perplexity_values

FUNCTION evaluate_lda_model(corpus, dictionary, processed_texts, start, limit, step):
    CALL compute_coherence_and_perplexity
    NORMALIZE coherence and perplexity scores
    COMPUTE composite scores
    FIND best model based on highest composite score
    PRINT best model details
    PLOT evaluation metrics
    RETURN best_model, best_num_topics, best_coherence, best_perplexity

FUNCTION plot_evaluation_metrics(start, limit, step, coherence_values, perplexity_values,
composite_scores):
    CREATE plot with three y-axes
    PLOT coherence scores
    PLOT perplexity scores
    PLOT composite scores
    ADD labels and legend
    DISPLAY plot

FUNCTION print_lda_topics(lda_model, num_words):
    FOR EACH topic in lda_model:
        PRINT topic words and weights

FUNCTION ensure_json_serializable(data):
    RECURSIVELY convert non-JSON-serializable data types to serializable types
    RETURN JSON-serializable data

FUNCTION clean_vis_data_in_place(vis_data):
    CLEAN 'Freq' column in vis_data.token_table using remove_complex_numbers
    CLEAN 'Freq' column in vis_data.topic_info using remove_complex_numbers
    CLEAN 'x' column in vis_data.topic_coordinates using remove_complex_numbers
    CLEAN 'y' column in vis_data.topic_coordinates using remove_complex_numbers

FUNCTION remove_complex_numbers(data_column):
    INITIALIZE empty cleaned_column
    FOR EACH value IN data_column:
        IF value IS complex number:
            ADD real part of value to cleaned_column
        ELSE:
            ADD value to cleaned_column
    RETURN cleaned_column

FUNCTION visualize_lda_model(lda_model, corpus, dictionary, notebook):
    PREPARE LDA visualization data
    IF in notebook environment:
        DISPLAY visualization in notebook
    ELSE:
        SAVE visualization as HTML file
        OPEN HTML file in web browser
```

FUNCTION get_top_ngrams(processed_texts, n, top_k):
    GENERATE n-grams from processed texts
    COUNT frequency of n-grams
    RETURN top_k most common n-grams

FUNCTION save_translated_texts(texts, filename):
    SAVE texts to pickle file

FUNCTION load_translated_texts(filename):
    LOAD texts from pickle file
    RETURN loaded texts

MAIN PROCESS:
    IF translated_texts_file exists:
        LOAD translated texts from file
    ELSE:
        text_data = extract_text_from_pdfs(pdf_directory)
        translated_texts = translate_text(text_data)
        SAVE translated_texts to file

texts = process_texts(translated_texts_file)
CREATE dictionary from texts
CREATE corpus from texts and dictionary

best_model, best_num_topics, best_coherence, best_perplexity = evaluate_lda_model(corpus, dictionary, texts)

PRINT details of best model

VISUALIZE best LDA model

FOR n FROM 2 TO 5:
    COMPUTE and PRINT top 5 n-grams

7 Word2Vec + K-means

The model which combined word2vec and k-means was based on the tutorial given by Castillo (2018). The model is described in Algorithm 6 and its respective code can be found in file word2vec_final_version.ipynb.

Algorithm 6 Topic Modelling with Word2Vec and K-means

SET seed to 23157003
INITIALIZE random number generators with seed

DOWNLOAD NLTK resources

SET pdf_folder_path to location of PDF files

DEFINE extra_stop_words
COMBINE extra_stop_words with Portuguese stopwords from NLTK

FUNCTION extract_text_from_pdf(pdf_path):

```
    INITIALIZE empty text string
    TRY:
        OPEN pdf_file
        FOR EACH page in pdf_file:
            ADD page text to text string
    CATCH any exceptions:
        PRINT error message
    RETURN text string

FUNCTION preprocess(text):
    CONVERT text to lowercase
    REMOVE numbers from text
    REMOVE punctuation from text
    TOKENIZE text
    RETURN list of tokens not in stop_words and longer than 2 characters

FUNCTION evaluate_word2vec(model, words):
    FOR EACH word in words:
    TRY:
        FIND most similar words using model
        PRINT similar words and their scores
    CATCH KeyError:
        PRINT word not in vocabulary

FUNCTION vectorize(list_of_docs, model):
    FOR EACH document in list_of_docs:
        INITIALIZE zero vector
        FOR EACH token in document:
            IF token in model vocabulary:
                ADD token vector to document vectors
        IF document vectors not empty:
            COMPUTE average of document vectors
        ELSE:
            USE zero vector
    RETURN list of document vectors

FUNCTION mbkmeans_clusters(X, k, mb, print_silhouette_values):
    PERFORM MiniBatchKMeans clustering
    COMPUTE silhouette score
    IF print_silhouette_values is True:
        COMPUTE and PRINT silhouette values for each cluster
    RETURN clustering model, cluster labels, and average silhouette score

FUNCTION find_best_num_topics(vectorized_docs, min_topics, max_topics, step, mb):
    INITIALIZE best_k and best_silhouette
    FOR k FROM min_topics TO max_topics STEP step:
        PERFORM clustering with k clusters
        IF silhouette score is better than best_silhouette:
            UPDATE best_k and best_silhouette
    PRINT best number of clusters and its silhouette score
    PLOT silhouette scores vs number of clusters
    RETURN best_k

FUNCTION plot_word_embeddings(model, words):
    FILTER words in model vocabulary
```

GET word vectors
PERFORM t-SNE dimensionality reduction
PLOT words in 2D space

FUNCTION plot_clusters(vectorized_docs, cluster_labels):
PERFORM t-SNE dimensionality reduction on vectorized_docs
PLOT documents in 2D space, colored by cluster labels

MAIN PROCESS:
INITIALIZE empty list for processed_texts

FOR EACH PDF file in pdf_folder_path:
text = extract_text_from_pdf(file)
processed_text = preprocess(text)
ADD processed_text to processed_texts

TRAIN Word2Vec model on processed_texts

EVALUATE Word2Vec model with sample words

vectorized_docs = vectorize(processed_texts, word2vec_model)

best_num_topics = find_best_num_topics(vectorized_docs, min_topics=2, max_topics=51, step=1, mb=100)

PERFORM clustering with best_num_topics

CREATE DataFrame with original texts, tokens, and cluster labels

PLOT word embeddings for sample words

PLOT clusters

PRINT top terms per cluster based on centroids

PRINT most frequent terms per cluster

ANALYZE and PRINT representative documents for a sample cluster

SAVE Word2Vec model

8 BERTopic

The BERTopic model is detailed in Algorithm 7. Its respective code can be found in the file named bertopic_final_version.ipynb.

Algorithm 7 Topic Modelling with BERTopic

SET seed to 23157003
INITIALIZE random number generators with seed

DOWNLOAD NLTK resources (stopwords and punkt)

```
SET pdf_path to location of PDF files

FUNCTION extract_text_from_pdf(pdf_file):
        INITIALIZE empty text string
        TRY:
                OPEN pdf_file
                FOR EACH page in pdf_file:
                        ADD page text to text string
        CATCH any exceptions:
                PRINT error message
        RETURN text string

FUNCTION preprocess_text(text):
        CONVERT text to lowercase
        REMOVE numbers from text
        REMOVE punctuation from text
        TOKENIZE text into words
        REMOVE stopwords from tokens
        RETURN joined tokens as string

FUNCTION compute_coherence_score(model, texts, topics, top_n_words):
        EXTRACT top words for each topic
        CREATE dictionary from texts
        COMPUTE coherence score using c_v measure
        RETURN coherence score

FUNCTION plot_all_topics(df, topic_model):
        COMPUTE mean positions of each topic
        GENERATE colors for topics
        CREATE scatter plot of all topics
        ANNOTATE top 50 topics
        ADJUST text annotations to avoid overlap
        DISPLAY plot

FUNCTION plot_top_10_topics(df, topic_model):
        IDENTIFY 10 most frequent topics
        FILTER DataFrame for top 10 topics
        COMPUTE mean positions of each topic
        GENERATE colors for topics
        CREATE scatter plot of top 10 topics
        ANNOTATE top 10 topics
        ADJUST text annotations to avoid overlap
        DISPLAY plot

FUNCTION visualize_hierarchy_custom(topic_model, topics, top_n_topics, width, height):
        IF topics not provided:
                GET all topics from model
        IF top_n_topics specified:
                CREATE labels for each topic
        LIMIT topics to top_n_topics
        EXTRACT topic embeddings
        PERFORM hierarchical clustering
        CREATE labels for topics
        GENERATE dendrogram
```

```
        DISPLAY plot

FUNCTION visualize_topics_barchart(topic_model, top_n_topics, n_words, width, height):
        GET topic information from model
        SORT topics by size (excluding outliers)
        SELECT top N topics
        CREATE labels for each topic
        GENERATE horizontal bar chart
        CUSTOMIZE plot appearance
        ADD count labels to bars
        DISPLAY plot

INITIALIZE empty list for texts

FOR EACH pdf_file in pdf_path:
        extracted_text = extract_text_from_pdf(pdf_file)
        IF extracted_text is not empty:
                preprocessed_text = preprocess_text(extracted_text)
                ADD preprocessed_text to texts list

DEFINE extra_stop_words
COMBINE extra_stop_words with Portuguese stopwords

CREATE CountVectorizer with combined stopwords
CREATE UMAP model for dimension reduction

INITIALIZE empty list for coherence_scores
SET topic_range to range from 2 to 50, step 2

FOR EACH num_topics in topic_range:
        CREATE BERTopic model with num_topics
        FIT model to texts
        COMPUTE coherence score
        ADD coherence score to coherence_scores list

PLOT coherence scores against number of topics

SET optimal_topics to number of topics with highest coherence score

CREATE final BERTopic model with optimal_topics
FIT final model to texts

SAVE final model

REDUCE dimensionality of embeddings to 2D using UMAP

ENSURE reduced_embeddings is 2D

CREATE DataFrame with reduced embeddings, topics, and text lengths

GENERATE visualizations:
   CALL plot_all_topics()
   CALL plot_top_10_topics()
   CALL visualize_hierarchy_custom()
   CALL visualize_topics_barchart()
```

9 Stop Words

The list of stop words used to customise the NLTK file is provided in Table 2. The model which performed LDA to the translated documents used the NLTK file for the English language. For this reason, the stop words list was customised using the translated version of the words.

Table 2: Extra Stop Words

Portuguese Words	English Words	Portuguese Words	English Words
acidentes	accidents	empresa	enterprise
acerca	about	errôneas	erroneous
acidente	accident	exclusivamente	exclusively
acordo	agreement	fabricante	manufacturer
aeronáutica	aeronautics	fatos	facts
aeronáuticas	aeronautical	figura	figure
aeronáutico	aeronautic	final	final
aeronáuticos	aeronautical	fins	purposes
aeronave	aircraft	futuros	future
aeronaves	aircraft	geral	general
agência	agency	glossário	glossary
ambos	both	havia	had
anac	anac	hipóteses	hipothesis
and	and	incidente	incident
anexo	attachment	induzir	induce
após	after	informações	information
aspectos	aspects	internacional	international
aviação	aviation	interpretações	Interpretations
avião	airplane	investigação	research
brasileiro	brazilian	investigações	investigations
cenipa	cenipa	junto	together
centro	center	matrícula	registration
civil	civil	modelo	model
conclusão	conclusion	momento	moment
conduzidas	conducted	nacional	national
conforme	according	nada	nothing
contribuintes	taxpayers	neste	this
contribuiu	contributed	nº	no
cuja	whose	nota	note
culpa	blame	nsca	nsca
dados	data	objetivo	objective
desconhecido	unknown	ocorrência	occurrence
deste	this	of	of
determinar	determine	organização	organization

Portuguese Words	English Words
organizacionais	organizational
outras	other
outro	other
outros	other
piloto	pilot
pode	can
poderá	can
poderá	can
possível	possible
possuia	had
prevenção	prevention
propósito	purpose
qualquer	any
realizar	accomplish
registros	records
relatar	tell
relatório	report
segurança	safety
seripa	seripa
simplificado	simplified

Portuguese Words	English Words
sipaer	sipaer
sobre	upon
suma	all
tampouco	either
ter	have
termos	terms
the	the
tipo	kind
to	to
totais	total
trazer	bring
últimas	last
últimos	last
uso	use
utc	utc
utilização	use
voo	flight
vôo	flight

References

Castillo, D. (2018) *How to Cluster Documents Using Word2Vec and K-Means*. Available at: https://dylancastillo.co/posts/nlp-snippets-cluster-documents-using-word2vec.html.