Reproducibility Review of: Exploring Urban Place Function through User-Generated Textual Content

Item	Value
Title	Exploring Urban Place Function through User-Generated Textual Content
Authors	Shahreen Muntaha Nawfee, Dr. Stef De Sabbata and Dr. Nicholas Tate
Ref. paper	Nawfee, S. M., De Sabbata, S., and Tate, N. J.: Exploring Urban Place Function through User-Generated Textual Content, AGILE GIScience Ser., 6, 6, https://doi.org/10.5194/agile-giss-6-6-2025, 2025.
Codechecker(s)	Ilya Ilyankou (https://orcid.org/0009-0008-7082-7122)
Date of Check	2025-04-06
Summary	The manuscript demonstrated partial reproducibility with three of six figures recreated successfully, albeit with some variations.
Repository	https://github.com/reproducible-agile/Exploring-Urban-Place-Function-through-User-Generated-Textual-Content
Ref. certificate	Https://doi.org/10.17605/OSF.IO/BGY83

Table 1: Reproduction metadata

Output	Comment
btm_osm_kc_agile2.png	Figure 2 in the manuscript (“The keyword composition of functions generated by the Biterm Topic Model on the OSM POI tags”)
btm_wiki_kc_agile.png	Figure 2 in the manuscript (“The keyword composition of functions generated by the Biterm Topic Model on the Wikipedia Article Summaries”)
figure5.png	Figure 5 in the manuscript (“Keyword distribution per Functions from BERTopic Model on Wikipedia article Summaries”)

Table 2: Summary of output files generated

Summary

The manuscript was partially reproducible, with three of six figures recreated with some deviations, likely due to unset random states in the computational process. The remaining figures and tables could not be reproduced in this process due to code issues; the GitHub repository contained 16 code files but lacked sufficient documentation for navigating the complex workflow. While the code can be run on standard hardware with common Python and R libraries, there are no instructions on setup or library versioning. Communication with the authors (17 emails over ~10 days) was necessary to obtain intermediate data files and clarify the process.

Reproducibility reviewer notes

The reproducibility review began on 27 March and concluded on 6 April 2025. During that period, 17 emails were exchanged between the code checker and the corresponding author.

The manuscript contains seven Figures (one being a methodological flowchart and as such not meant for reproducibility) and three Tables (one containing LLM prompts and not meant to be reproduced). In this reproducibility review, I was able to partially reproduce three Figures (Figure 2, Figure 3, and Figure 5) that deviate from those in the manuscript. **This is likely because random states were not set throughout the process** (e.g., initializing the UMAP model and passing it to BERTopic, or calling the BTM function in R without setting the random seed first).

I was unable to continue running the code halfway past the 'wiki+osm_kc_new.ipynb' notebook due to code errors, and as such I was unable to generate other figures and tables.

The original GitHub repository accompanying this manuscript contained 10 Jupyter notebook files (.ipynb), four Python files (.py), and two R markdown files (.Rmd). Initially the README file did not contain any information on the sequencing of files to run, or expected inputs and outputs for each script; the README was later updated with some instructions but remains insufficiently detailed for such a long and complex process.

The code lacks comprehensive documentation.

Installation prerequisites and computational environment

The process can be run locally on a medium-powered laptop with Python, Jupyter, and R/RStudio installed. Several notebooks appeared to be written to run in Google Colab, but can be transformed into locally-run Jupyter notebooks by removing '`from google.colab import files`', and '`uploaded = files.upload()`', and reading CSV files directly into the data frames instead.

The Python libraries are standard for geospatial and scientific computing, and include `pandas`, `geopandas`, `shapely`, `bertopic`, `numpy`, `matplotlib`, `sklearn`, `scipy`, `itertools`, `gensim, io, umap`. The R packages include `tidytext`, `tidyverse`, `BTM`, `textplot`, `ggraph`.

There are no installation or setup instructions regarding which versions of libraries and packages to use. The code appears to rely on an older version of pandas; for example, the `df.append()` method was deprecated in v2.0 (April 2023) and as such there was an error when running on a newer version of `pandas`; code amendment (replacing `df.append()` with `pd.concat())` was necessary to run the process.

There are several attempts to install Python packages on the fly using '`pip install <library>`' without specifying the library version, which may eventually lead to incompatibilities.

Data preparation

The project relies on two data inputs, (1) a Wikipedia SQL dump and (2) geographic boundary for the Kensington & Chelsea borough.

Documentation to obtain and process the input files was created upon the code checker's request.

Converting SQL into CSV involves 11 steps as per README, and requires a locally run MySQL database system. However, the SQL dump in the format expected by the notebook appears to no longer exist, which makes replication difficult; an intermediate CSV file representing the `kc_wiki_clip` dataframe was requested and received by the code checker from the corresponding author during the review.

Both sources are publicly accessible with permissive licenses, and ideally should be part of the GitHub repo or ingested and processed directly using tools such as `wget, requests` or `geopandas`.

Running the code

The sequencing of files to run is now documented in the GitHub's README. Jupyter notebooks can be run in several minutes each, and most output CSV and image files.

The sheer *number* of notebooks, their naming (e.g., '*osm_kc_dbscan_lda_btm_new2.ipynb*', '*kc_wiki_summary_lda_btm_new2.ipynb*') , and the output file structure and naming (e.g., '*../../docs/docs_agile/topic_prob_kc_osm.csv*') make it somewhat difficult to keep track of the process.

There were several occasions of loading a shapefile ('*data/boundary_london/kc.shp*') that didn't come with instructions on how to get it (I replaced the Shapefile with the *'kc_2011_oa.geojson'* file to be able to continue). I also ran into an issue with mis-matched counts when assigning geometries to a geodataframe in *osm_kc_dbscan_lda_btm_new2.ipynb* (replaced 99 with 101 in a 'for' loop).

Outputs and results

Figure 2 – Manuscript

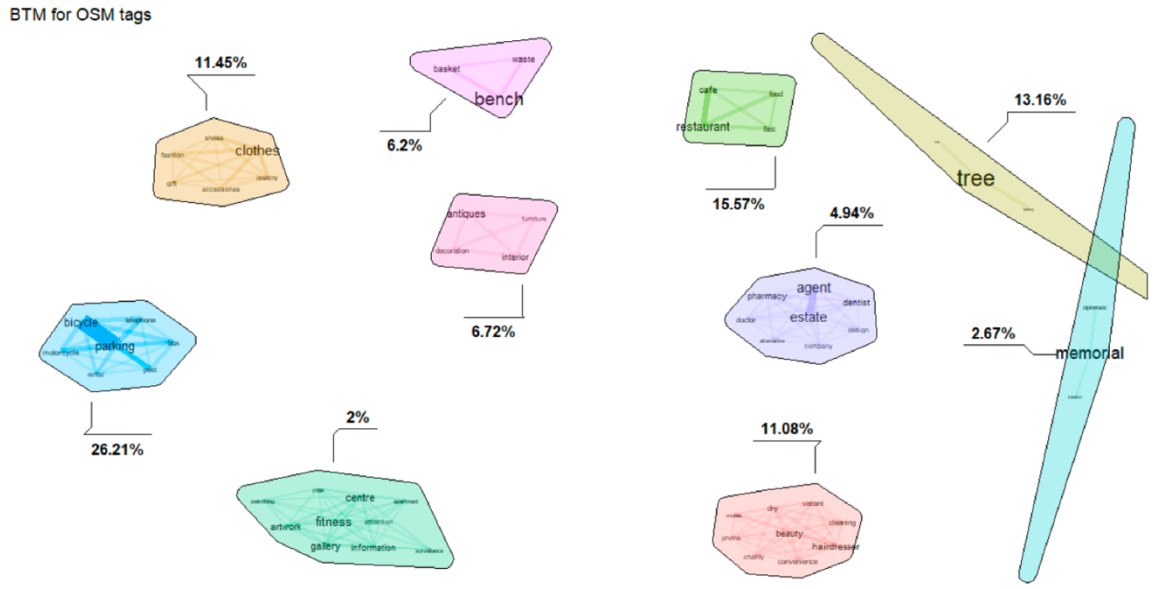


Figure 2. The keyword composition of functions generated by the Biterm Topic Model on the OSM POI tags

Figure 2 – Generated

The generated figure differs in the number and contents of clusters (8 vs 10 in the original Figure 2), although some clusters are comparable (e.g. the original blue cluster with “parking” and “bicycle” is the same as purple in the reproduced Figure).

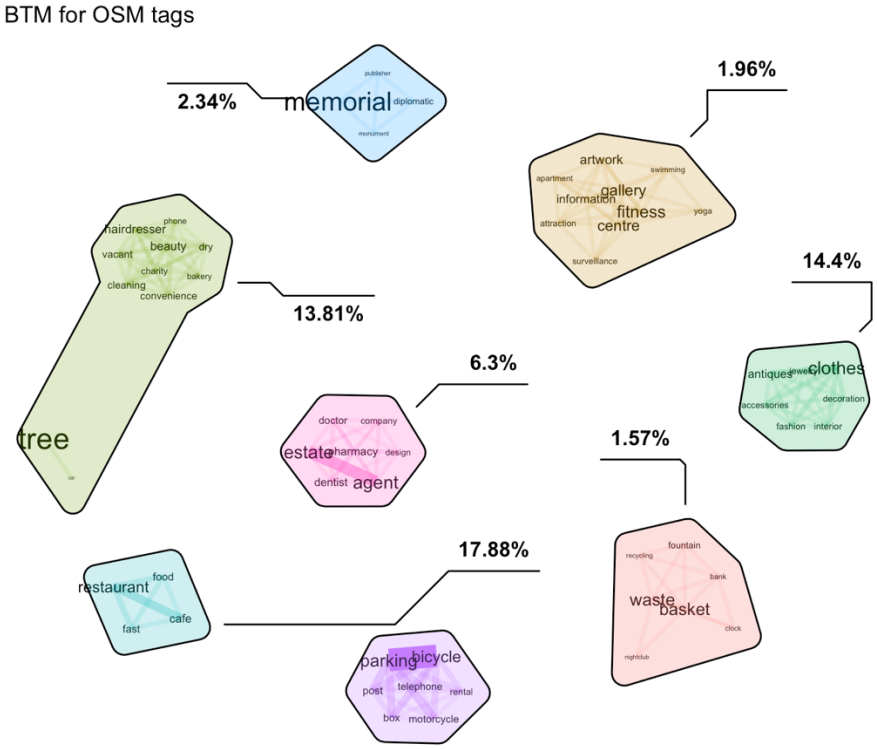


Figure 3 – Manuscript

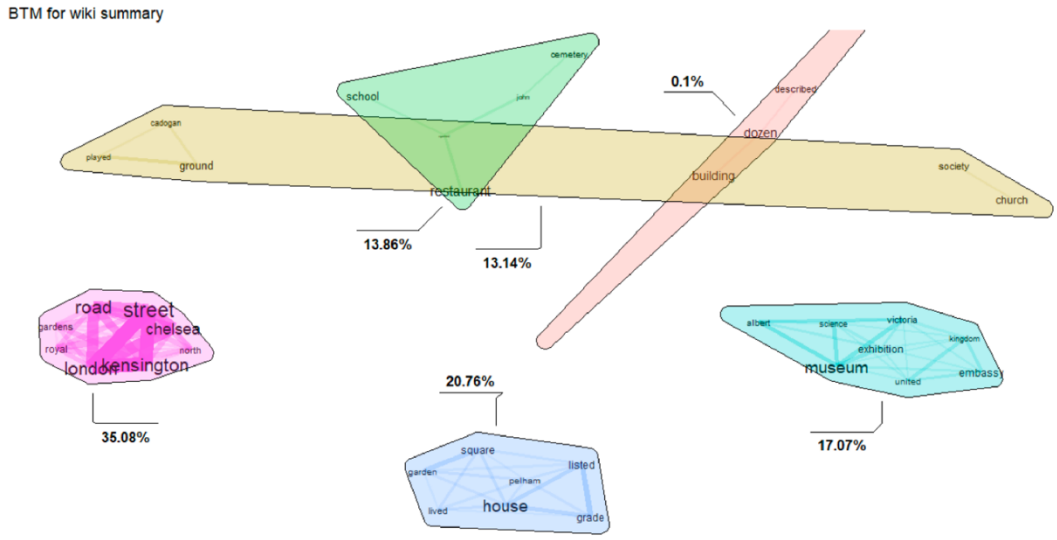


Figure 3 The keyword composition of functions generated by the Biterm Topic Model on the Wikipedia Article Summaries

Figure 3 – Generated

The figure differs in style and the number of clusters to a point where it's unclear whether the same code was used to produce the original Figure 3.

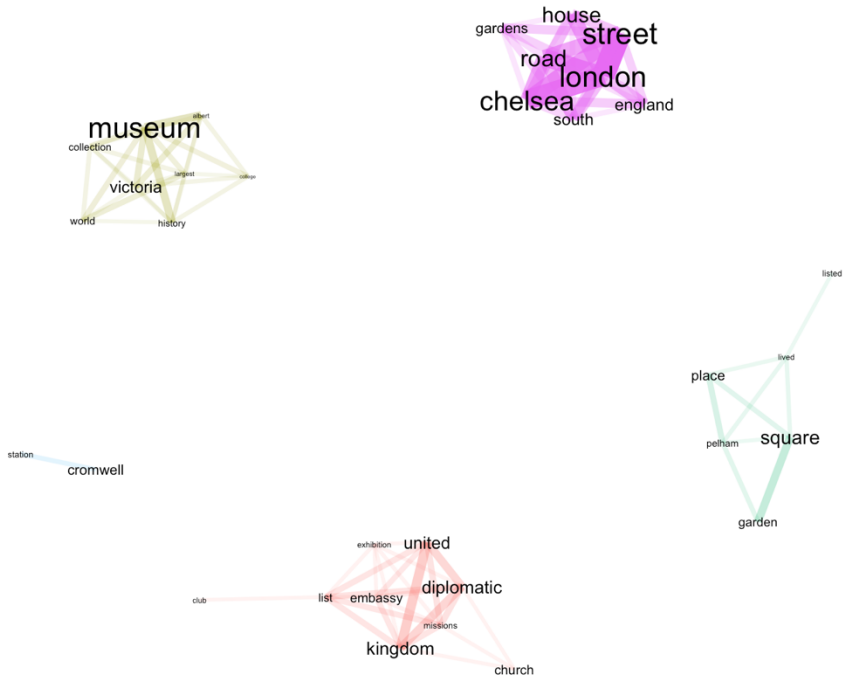
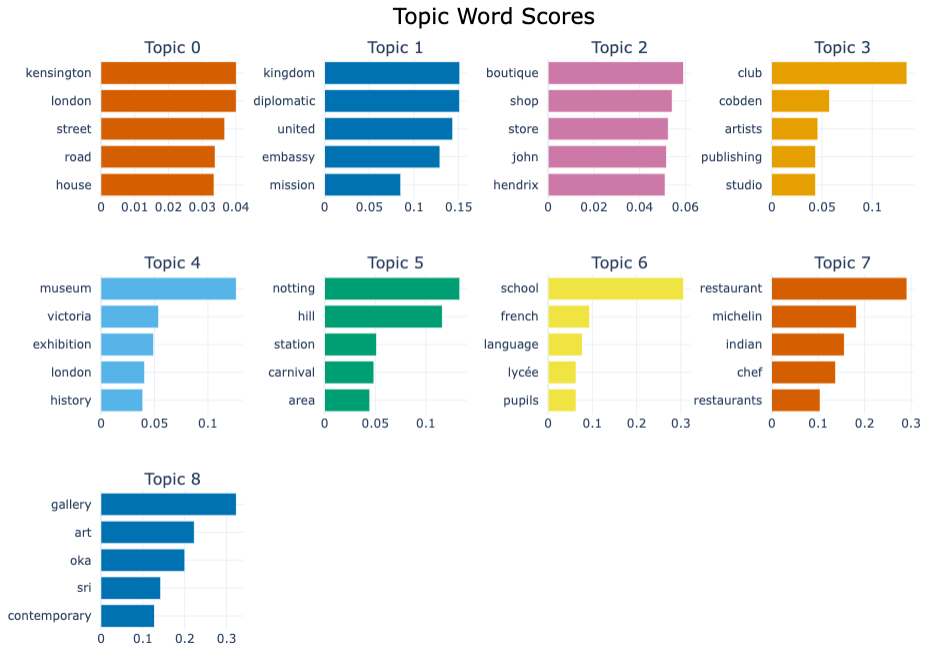


Figure 5 – Manuscript

Figure 5. Keyword distribution per Functions from BERTopic Model on Wikipedia article Summaries

Figure 5 — Generated

The reproduced bar chart contains 8 Topics vs 6 in the manuscript's version. The ordering (numbering) of Topics, as well as ordering of bars within Topics differs, although some can be matched (e.g., manuscript's Topic 2 is the reproduced figure's Topic 1).



Recommendations

1. Format and expand README so it's easier to understand the process and see key results (figures and tables)
2. Consider using a virtual environment, and specify Python library versions (e.g. use `venv` or `conda` with `requirements.txt` - https://www.geeksforgeeks.org/how-to-create-requirements-txt-file-in-python/)
3. Consider consolidating the process from current ~16 files into fewer (or even a single) notebooks
4. Use Markdown cells to explain the process and document the code inside the notebooks
5. Ingest and clean input data programmatically where possible; avoid manual downloads, edits, and format conversions for efficient reproducibility
6. Simplify the outputs; e.g. consider creating a single '`output`' folder with clearly labelled files such as 'fig2.png'

Citing this document

This report is part of the reproducibility review at the AGILE conference. For more information see https://reproducible-agile.github.io/. This document is published on OSF at OSF DOI https://doi.org/10.17605/OSF.IO/BGY83.

To cite the report, use:

Ilyankou, I. (2025, April 22). Reproducibility review of: Exploring Urban Place Function through User-Generated Textual Content. Retrieved from osf.io/bgy83