ISyE 6740 – Fall 2021
Strava Segment Analysis with Latent Dirichlet Allocation

Andrew Traylor (902815209)

Problem Statement

Presently, a multitude of cyclist-tracking-apps exist which provide riders with summary information such as GPS route, average speed, elevation change, and trip duration. The most popular of such activity-tracking apps, Strava, facilitates activity sharing among users, if desired. Perhaps Strava's most widely used feature is the creation and tracking of user-defined "Segments," portions of a path or road which the user has traveled. Riders can create these segments from their activities, upload them to the Strava platform, and compare times with other users.

Strava also provides a "Routes" feature which allows riders to create routes based on three criteria:

1) **Distance**. An approximation of how far the rider would like to travel.
2) **Elevation**. An option to avoid hills or add elevation gain.
3) **Surface**. Optimize for mostly paved or mostly dirt routes.

The Strava algorithm generates these routes based on "popular waypoints", in addition to the filtering criteria above. The full web application also provides the option to select the most direct route, indifferent to the above criteria.

By generating recommended routes based on the most trafficked ride "segments," Strava likely recommends routes that do not optimize for rider safety or comfort. Furthermore, certain users may have different revealed "comfort" preferences such as avoiding intersections, traffic, etc. that differ from the average Strava rider.

Given that a rider's route can be viewed as a string of segments, a more thoughtful recommender might take into account segment nuance, rather than relying solely on segment density as a means of assessment. There may be a subset of riders less concerned with fitness and/or competition with peers, and more concerned with the features of the segments that compose the route.

This paper will attempt to use topic modeling, a technique common in natural language processing, to identify groups of segments which may be lost in simple popularity, distance, elevation, and surface filters.

Data Source

"Strava Metro" is a program which displays aggregated user data to help urban planners and other civic groups understand cities' transportation needs. Strava Metro has been made available

to state and city governments for free, but at the time of writing, access for other individuals remains restricted. Therefore, the publicly available Strava API was used to extract segment data.

US cities were first split into three bins based on population counts provided by the US Census Bureau:

- Large: Greater than 1,000,000
- Mid-size: Between 100,000 - 1,000,000
- Small: Less than 100,000

To reduce the effects of one size or style of city dominating the analysis, seven cities were randomly sampled from each size category for segment retrieval. A full list of the sampled cities can be found in **Figure 1** of the Appendix.

The Strava API endpoint of interest returned the top ten most-travelled segments within a bounding box described by geographic coordinates. Additionally, the Strava API maintains 15-minute and daily request limits of 100 and 1,000 requests, respectively. The segment retrieval problem therefore became one of obtaining the most segments, per city, within the constraints of rate usage.

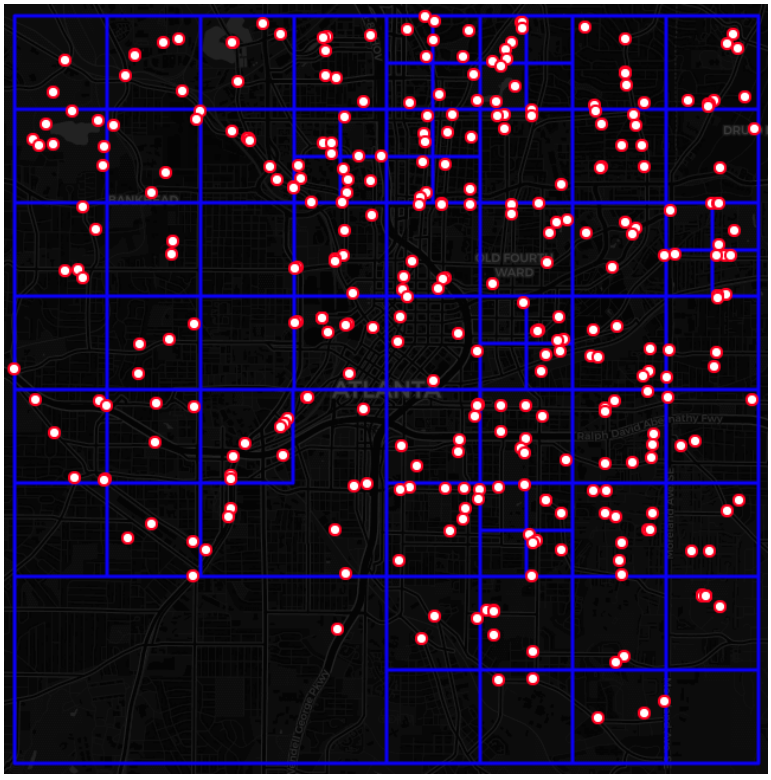
Rather than building a 10x10 grid around the city center (100 API requests), which might result in empty or nearly-empty queries, or building concentric boxes outward from the city center, quadtrees were constructed for each location of interest. A quadtree, or in this case, a point-region quadtree, is a tree data structure that partitions an area into four subspaces, each with some max capacity. If the subspace (node) contains more than the max number of points, it is further subdivided into four nodes. Once a node contains fewer than the max capacity of points, it is no longer partitioned and becomes a leaf.

Therefore, in pseudo-code, the quadtree recursion steps for Strava API segment retrieval were as follows:

For each city:

1) Find the city center
2) Build a 25-square-mile bounding box around this center
3) Query the API for the top ten segments
4)
    i) If the number of segments returned is ten (the maximum per the API call):
        divide the box into four smaller boxes.
    ii) Else:
        end the query.
5) Repeat until the API limit is reached.

For example, this algorithm returned 332 Strava segments for the city of Atlanta, Georgia, as represented in quadtree form here (endpoints of segments shown):



After a line was obtained for each segment, buffers were added around the lines using the GeoPandas python library, such that the segments were transformed into queryable polygons; inherently, lines cannot contain features, but polygons can. For example, here are the segments (red) and "buffered" segments (blue, enlarged for illustration) for the city of Atlanta:

For each buffered segment, the OpenStreetMap database was queried using the python library OSMnx. OpenStreetMap is an open-source project analogous to a geographic wikipedia which contains millions of user-uploaded key-value pairs known as tags. These tags describe various features of elements found on a map. For example, the "highway=residential" tag describes a road which runs along a residential area. Only tags with at least 50,000 entries in the OSM database were queried. The result of the OSMnx queries was a count of occurrences of tags for each segment. For example, the segment with ID 27581807 contained three tags of "surface=concrete", indicating a concrete surface at three various "ways" within the segment.

The final data matrix contained 1,050 rows (representing segments), and 399 columns (representing counts of OSM tags).

"Topic" Modeling and tf-idf

After extraction of the raw tag counts, the parallels between a) segments and counts of tags, and b) documents and counts of words became evident. Therefore, the general approach for finding similarity among Strava segments became that of NLP-esque topic modeling. Topic models are built around the idea that, hidden between documents and the words they contain, exists a middle layer of topics. Documents contain topics, and topics contain words. Therefore, analysis can be conducted to uncover these "latent" or unobserved topics. Regarding Strava, it can be said that segments (documents) are composed of *k* segment types (topics), and these segment types have a different composition of OSM tags (words).

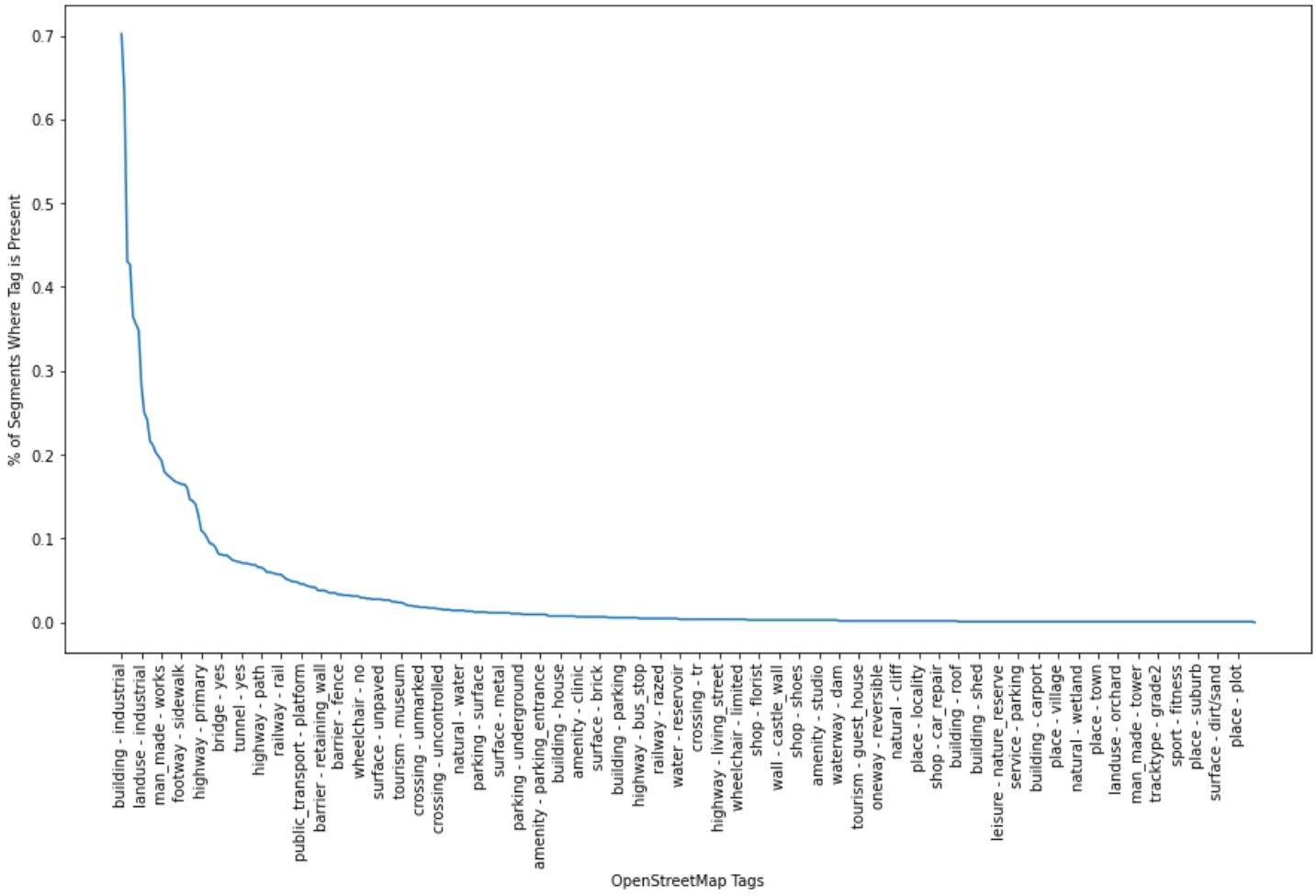
Term frequency-inverse document frequency (tf-idf) is commonly used in the field of natural language processing to scale document-term matrices. The general idea is to scale raw word counts so that the counts of words appearing across many documents are reduced, while the counts of words that appear infrequently across documents (so-called "rare" words) are increased. Tf-idf scales counts through multiplication with a constant (Ramos, 2003):

$$w_d = f_{w,\,d} * \log\left(|D|/f_{w,\,D}\right)$$

where w_d is the tf-idf scaled count, $f_{w,d}$ is the raw count, D is the number of documents in the corpus, and $f_{w,D}$ is the number of documents in which the word appears.

Tf-idf helps to reduce outlier counts that might skew correlation and variance statistics when performing dimensionality reduction, such as principal components analysis, or which might affect similarity metrics within certain techniques like k-means clustering. An additional advantage of tf-idf is the retention of sparsity in the data matrix; other common scaling methods such as min-max-scaling and standard scaling require either normalization or centering of the data, transforming sparse sparse matrices into dense matrices and subsequently introducing computational overhead. Worse, traditional scaling methods may introduce nonzero values for terms (tags) that were not previously present in a document (segment).

The Strava data matrix lends itself nicely to tf-idf scaling. It is heavy-tailed in that some tags appear in a large percentage of segments, while others appear only once or twice:



An attempt at Latent Semantic Analysis (LSA)

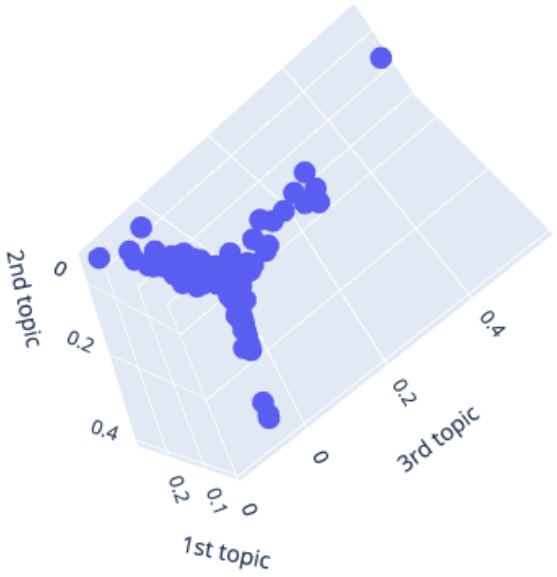
In natural language processing, latent semantic analysis is the performance of truncated singular value decomposition (SVD) on a document-term matrix to uncover the hidden topics within a document. The document-term matrix consists of redundant, noisy features, so the general idea is to project the features into a lower-dimensional space, where the number of dimensions represents the number of topics. An advantage of using truncated SVD over traditional SVD is the preservation of sparsity in the tf-idf data matrix, since traditional SVD requires centering of the data. The principal components post-SVD represent the latent topics that rest in between the document (segment) and the words (tags). Optionally, metrics such as cosine similarity can be computed to assess document similarity. In this paper, semantic analysis is a misnomer, but the analogy holds well.

In the context of Strava, the input of the LSA is the segment-tag matrix whose decomposition through truncated SVD results in the following three matrices:

$$M = U * \Sigma * V^T$$

Here, M represents the original tag X segment matrix, U represents the tag X topic matrix (projection of tags into topic space), Σ represents singular values of the topics, and V^T represents the topic X segment matrix. To keep topics as the columns, V can be analyzed as the segment X topic matrix.

The most glaring problem with the truncated SVD output is its lack of interpretability. The topics are hardly intuitive, and both the tag weightings within topics and the topic weightings within segments are arbitrarily positive or negative. For example, projecting each segment into a three dimensional-topic space results in the following geometric interpretation



Where each segment has a set of topic (component) loadings such as:

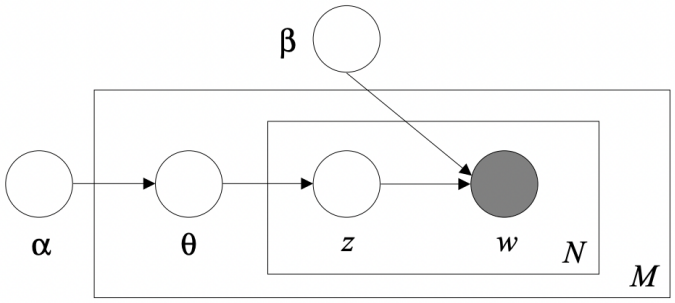
Segment	1st topic	2nd topic	3rd topic
0	0.0011	0.0005	0.0008
1	0.0211	0.0121	-0.0018
2	0.002	0.0033	0.0001
3	0.0682	-0.0115	-0.0108
4	0.0054	0.0071	0.0006

While traditional clustering approaches such as k-means or Gaussian mixture models could have been used in this new lower-dimensional space to group segments, or a cosine similarity approach as noted above, a lack of interpretability ultimately pushed the analysis away from an SVD-based approach.

An attempt at Latent Dirichlet Allocation (LDA)

The central idea behind latent Dirichlet allocation in the context of NLP is that documents can be described by a distribution of topics, and, in turn, each topic can be described by a distribution of words.

The process for generating a document, w, within a corpus (collection of documents, M) can be visualized as follows, from the original latent Dirichlet allocation paper (Blei et al., 2003):



where the outer plate represents documents across the entire corpus (M), and the inner plate, N, represents topics (z) and words (w) within those documents. θ represents a Dirichlet distribution, tuned by α. β represents the Dirichlet distribution of words within a topic.

Alpha and beta may be thought of as "concentration" parameters that control the spread of the Dirichlet distributions; by controlling alpha and beta one controls topic and word sparsity within a document, respectively. For example, an alpha < 1 will result in a topic mixture that contains a few, if not one, topic, while an alpha > 1 will generate a topic mixture that contains most, if not all, topics.

LDA is a generative, probabilistic model, designed to reproduce documents that resemble the original corpus using the α, β, and θ parameters learned from the training process. In practice, the goal is rarely to generate new documents. Rather, the aim is to infer the probabilistic composition of both documents and topics by maximizing the likelihood of word occurrence across a corpus using the document-topic distributions and the topic-word distributions. More mathematically, the objective is to find parameters α and β that maximize the (marginal) log likelihood of word occurrence across the corpus (Blei et al., 2003):

$$\ell(\alpha, \beta) = \sum_{d=1}^{M} \log p(\mathbf{w}_d \,|\, \alpha, \beta).$$

This is typically done through a variational expectation-maximization procedure, or through Gibb's sampling.

The output of the LDA algorithm is a distribution of topics within each document, and a distribution of terms within each topic. In the case of the Strava data matrix, the output of LDA

is a distribution of topics for each segment, and a distribution of OSM tags within each of those topics. The most notable drawback of LDA is that the topics typically require human interpretation and labelling.

Evaluation and Final Results

For LDA, alpha and beta are key hyperparameters of consideration during model training, as is the number of topics. As such, a grid search was conducted using the LatentDirichletAllocation class within scikit-learn with the following combinations of hyperparameters:

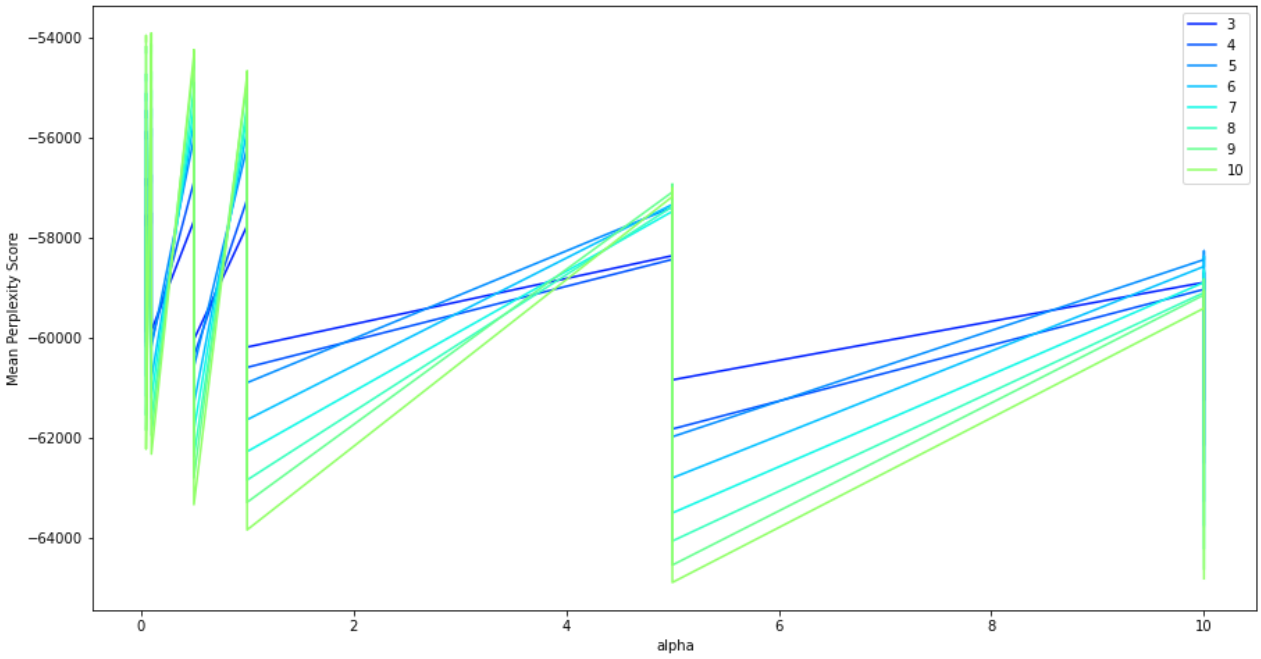
- Number of topics: 3-10
- Document topic prior (α): [0.05, 0.1, 0.5, 1, 5, 10]
- Topic word prior (β): [0.05, 0.1, 0.5, 1, 5, 10]

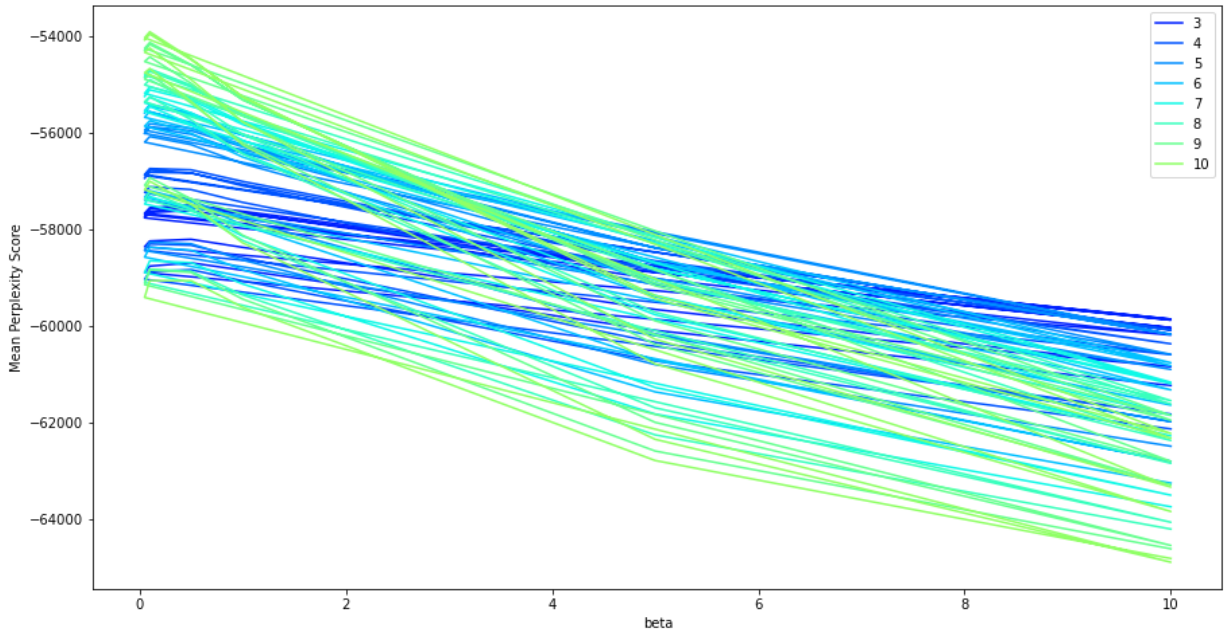
Models were compared on the basis of mean five-fold cross-validation "Perplexity", defined as follows for a set of M documents (Blei et al., 2003):

$$perplexity(D_{\text{test}}) = \exp\left\{-\frac{\sum_{d=1}^{M}\log p(\mathbf{w}_d)}{\sum_{d=1}^{M} N_d}\right\}.$$

A lower perplexity score implies a greater log likelihood of word occurrence across the corpus.

The best model had α and β parameters of 0.1, and the number of topics was selected to be 10. Lower alpha and beta values, along with higher topic counts, were associated with lower mean perplexity scores (topic count shown in legend):





See Figure 2. of the appendix for a list of parameters associated with the top ten training scores.

Regarding topic coherence, the tags appeared to explain a consistent theme for each topic. The ten types of Strava segments could be loosely viewed as centering on the following themes, after examining the probabilities of tags within each topic:

	Topic Themes
1	wooded/natural
2	minor access roads
3	buildings; restaurants; retail
4	bike and foot paths; benches; park likely
5	pedestrian crossings/sidewalks; asphalt/concrete surfaces; many traffic signals
6	curb/unmarked crossing/sidewalk
7	residential areas
8	well-lit, minor highways
9	well-lit, pedestrian walkways
10	minor highways with possible bus stops

See Figure 3. of the appendix for a visual of the top ten tags per topic, as represented by their probability of occurrence within each of the topics.

As for real-world coherence, further analysis is required to assess topic alignment with on-the-ground reality. For example, for the city of Seattle, it remains to be seen if the dominant topics indicated for each sampled segment actually contain those features (color by dominant topic):

Cyclist feedback for verification is paramount. However, it may be possible to validate the results by looking at the breakdown of topics by cities, and confirming that more or less bike-friendly cities contain a greater/lower percentage of certain segment types.

Potential areas of future research include:

- **Training the LDA model on a much larger number of Strava segments**. A relatively low proportion of features (121) were used as the segment OSM "vocabulary." While the analysis captured most of the relevant OSM tags, it could be expanded.
- **The effects of more extensive feature engineering prior to model training**, particularly as it pertains to combining rare Strava tags into categories with higher counts, perhaps aggregating by key.
- **Evaluating model hyperparameters through topic coherence scores** rather than "perplexity."
- **The development of a better route recommender**. The current Strava "recommend by route density" algorithm does not account for segment nuance and the probability of certain features being present on a route.

In conclusion, the research appears promising that latent Dirichlet allocation could assist in the identification of safer and perhaps more pleasurable Strava segments, beyond the information provided by simple filtering metrics such as popularity, distance, and terrain. LDA probabilistic interpretations of segment OpenStreetMap tags could be incorporated in future recommendation algorithms to help improve the experience of cyclists, particularly those less representative of the average Strava rider.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, 993-1022.

Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).

Appendix

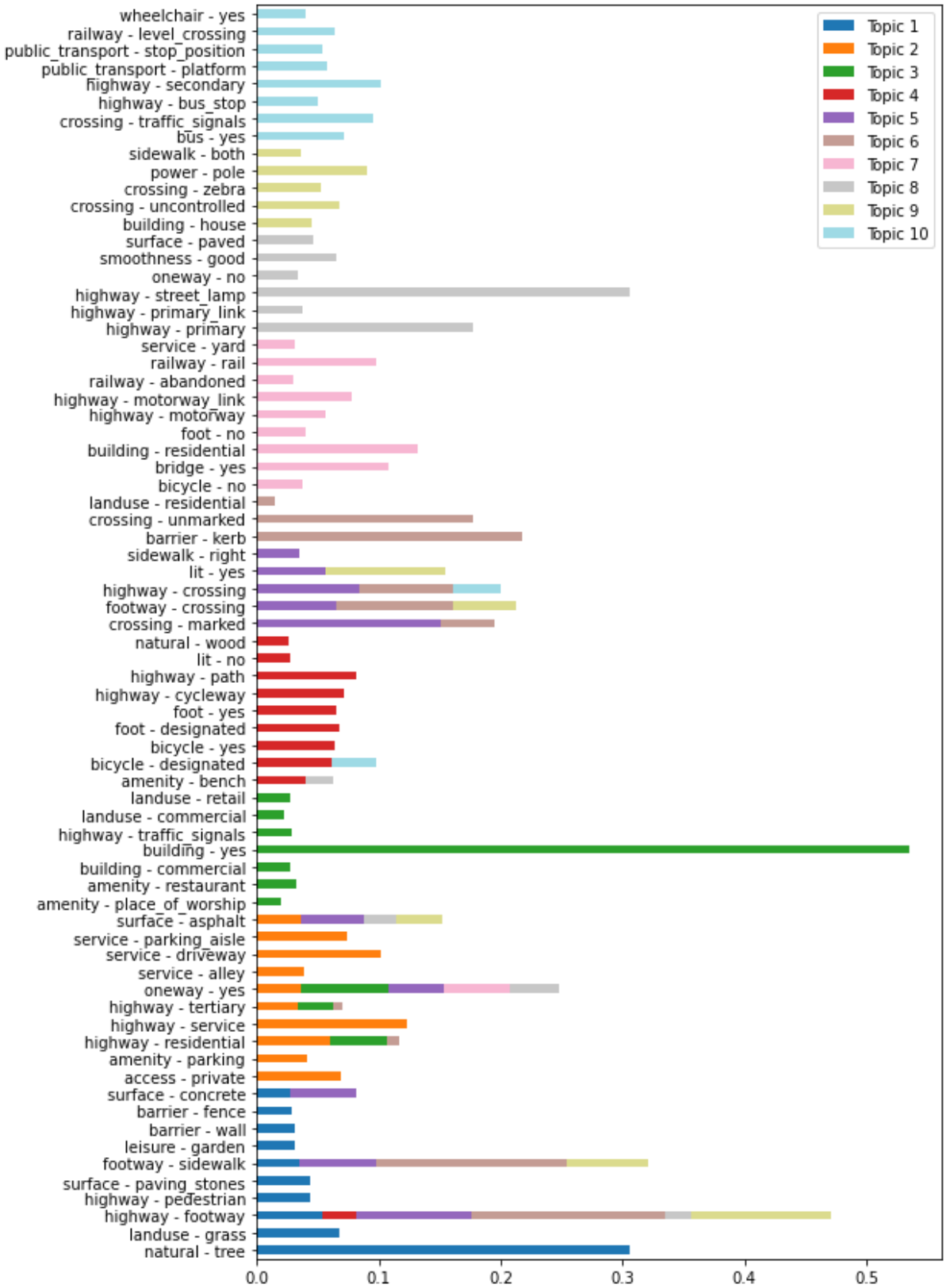
Figure 1.

NAME	POP
Philadelphia, PA--NJ--DE--MD	**5441567**
Atlanta, GA	**4515419**
Seattle, WA	**3059393**
St. Louis, MO--IL	**2150706**
San Antonio, TX	**1758210**
Pittsburgh, PA	**1733853**
Jacksonville, FL	**1065219**
New Orleans, LA	**899703**
Tucson, AZ	**843168**
Albany--Schenectady, NY	**594962**
Colorado Springs, CO	**559409**
Charleston--North Charleston, SC	**548404**
Augusta-Richmond County, GA--SC	**386787**
Ann Arbor, MI	**306022**
Wenatchee, WA	**67227**
Saratoga Springs, NY	**64100**
Kingston, NY	**57442**
Cookeville, TN	**44207**
Lebanon--Hanover, NH--VT	**25690**
Laconia, NH	**18636**
Eureka, MO	**11260**

Figure 2.

param_doc_topic_prior	param_n_components	param_topic_word_prior	mean_test_score	std_test_score	rank_test_score
0.1	10	0.1	-53906.871616148100	20611.584326787700	1
0.05	10	0.1	-53955.57000490030	20685.471763993100	2
0.1	10	0.05	-54030.153947137200	20602.04556154840	3
0.05	10	0.05	-54075.17880072040	20667.07871014970	4
0.1	9	0.1	-54128.44189485400	20657.92582386890	5
0.05	9	0.1	-54169.380510404100	20725.550240689900	6
0.5	10	0.1	-54245.93529012620	20470.474193667300	7
0.1	9	0.05	-54249.42905169210	20643.69650255930	8
0.05	9	0.05	-54295.90245269950	20706.353829552300	9
0.5	10	0.05	-54332.387173288300	20452.48882179110	10

Figure 3.


wheelchair - yes
railway - level_crossing
public_transport - stop_position
public_transport - platform
highway - secondary
highway - bus_stop
crossing - traffic_signals
bus - yes
sidewalk - both
power - pole
crossing - zebra
crossing - uncontrolled
building - house
surface - paved
smoothness - good
oneway - no
highway - street_lamp
highway - primary_link
highway - primary
service - yard
railway - rail
railway - abandoned
highway - motorway_link
highway - motorway
foot - no
building - residential
bridge - yes
bicycle - no
landuse - residential
crossing - unmarked
barrier - kerb
sidewalk - right
lit - yes
highway - crossing
footway - crossing
crossing - marked
natural - wood
lit - no
highway - path
highway - cycleway
foot - yes
foot - designated
bicycle - yes
bicycle - designated
amenity - bench
landuse - retail
landuse - commercial
highway - traffic_signals
building - yes
building - commercial
amenity - restaurant
amenity - place_of_worship
surface - asphalt
service - parking_aisle
service - driveway
service - alley
oneway - yes
highway - tertiary
highway - service
highway - residential
amenity - parking
access - private
surface - concrete
barrier - fence
barrier - wall
leisure - garden
footway - sidewalk
surface - paving_stones
highway - pedestrian
highway - footway
landuse - grass
natural - tree
0.0
0.1
0.2
0.3
0.4
0.5
Topic 1
Topic 2
Topic 3
Topic 4
Topic 5
Topic 6
Topic 7
Topic 8
Topic 9
Topic 10