Student Name: Kai Hirota

Student ID: 500335538

UniKey: khir2693

Cluster Analysis of Airbnb listings using word embeddings on listing descriptions

I. Abstract

Airbnb is an online platform that allows travelers to find traditional and alternative accommodations (listings) to stay at. The platform has 7 million listings worldwide, and 36,662 in Sydney [1]. Each listing has a description, written by the host to provide essential information and unique selling points to persuade visits. To make the discovery faster and improve product usage, information that can be used to effectively segment or categorize listings should be extracted from listing descriptions, if any. We aim to discover clusters in Airbnb listings by training KMeans on the average word embedding of the listing descriptions. Specifics of word embeddings will be discussed in section lll.

II. Background and Research Problem

Research Question: Can we use word embeddings to discover clusters in Airbnb listings?

For this research, Sydney Airbnb Open Data from Kaggle will be used [2]. It contains 36,662 rows of listings data, with 96 columns, one of which being listing description. On average, a listing review has 133.19 words, with standard deviation of 52.93, suggesting that each listing will have more than enough words even after cleaning the data. The null hypothesis is that the average word embedding of a description cannot be used to cluster listings into groups that share certain attributes. The alternative hypothesis is that the average word embedding of a description can be used to cluster listings into groups that share certain attributes. To quantify the reliability and performance, we will use the silhouette score [3]. If there are underlying clusters, and the number of clusters is selected appropriately, then the silhouette score should be between 0 and 1. If the number of clusters is not selected appropriately, then the score will be between -1 and 0. If the score remains approximately flat and close to 0 regardless of the number of clusters, then there are no underlying clusters. V-measure will not be used since we have no labels to evaluate the clustering.

III. Approach

As the first step, since listing descriptions are not always written in English, langdetect, Google's language-detection library, was used to label the language each description is written in [4]. Any description written in languages other than English were removed. Then, listing descriptions were preprocessed using a Natural Language Processing library for Python called SpaCy. Stop words, or words that do not add any meanings to a sentence, as well as punctuations were removed from each listing description. All words were converted to lower-case and lemmatized as well. When a word is lemmatized, that word is converted to the present-tense and the most basic form of that word. For example, "lemmatizing" will be converted to "lemmatize". Then, using a feature called Named Entity Recognition, all the instances of named entities - "Bondi Beach", "John", "Ultimo", to name a few - were removed, since names are not included in pre-trained word embeddings.

Inverse document frequency was computed for every word that appears in the "document", which in this case refers to the listing descriptions.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

N: Total number of documents

$|\{d \in D : t \in d\}|$: Number of documents where the term t appears

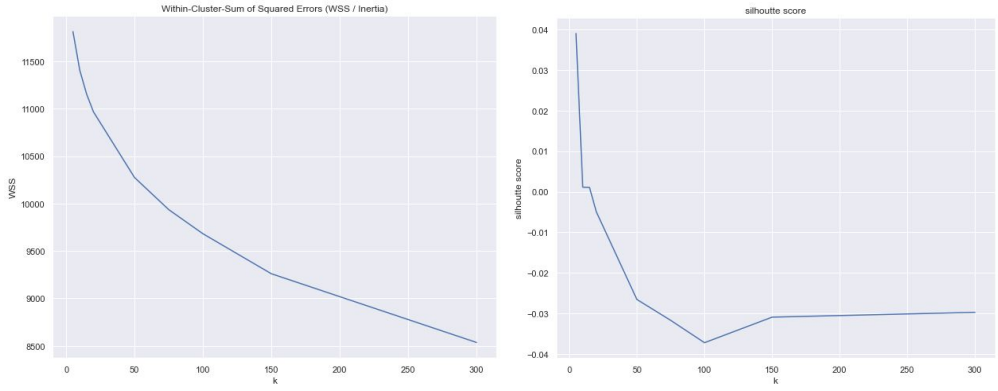
Inverse document frequency is the measure of the amount of information a term provides, with respect to the documents being analyzed. For example, if a given term appears in every listing description, then the idf will be 0. The rarer a term is, the higher the idf will be [5]. For this project, the threshold of 5% was chosen; terms that appear in 5% or more of the documents were removed. This translates to idf of 1.3, and removed a total of 230 words from listing descriptions, such as “good”, “great”, “perfect”, “access”, “toilet”, “train,” and so on.

After all of the above text preprocessing, we expect each listing description now contains mostly the unique words that differentiate each listing. There are now 35,154 rows left in the dataset, containing 22.43 words in the description on average, down from an average of 133.19 words per description before preprocessing. We then convert each description into word embeddings. Word embeddings is a tool for representing words such that words with similar meanings have a similar representation. Individual words are represented as real-valued vectors in a predefined vector space, and each vector is learned based on the usage of the word, “allowing words that are used in similar ways to result in having similar representations, naturally capturing their meaning” [6]. In this project, we will be using one of the pre-trained English word embeddings trained using fastText, made publicly available by Facebook’s AI Research Lab [7], consisting of 1 million word vectors with 300 dimensions. For each listing description, we look up the word embedding vector for each term remaining in the description and take the mean of all the vectors.

To reiterate, the alternative hypothesis is that these “average word embedding” of each description should preserve the uniqueness of the listing, and should allow us to discover hidden clusters in listings using a clustering algorithm such as KMeans. After converting listing descriptions to “average word embeddings,” we now have a flat, 300-dimensional vector for each listing description.

IV. Results

KMeans was used for clustering the average word embeddings. To evaluate the clustering performance, inertia and silhouette score were calculated for various k values: [5, 10, 15, 20, 50, 75, 100, 150, 300].



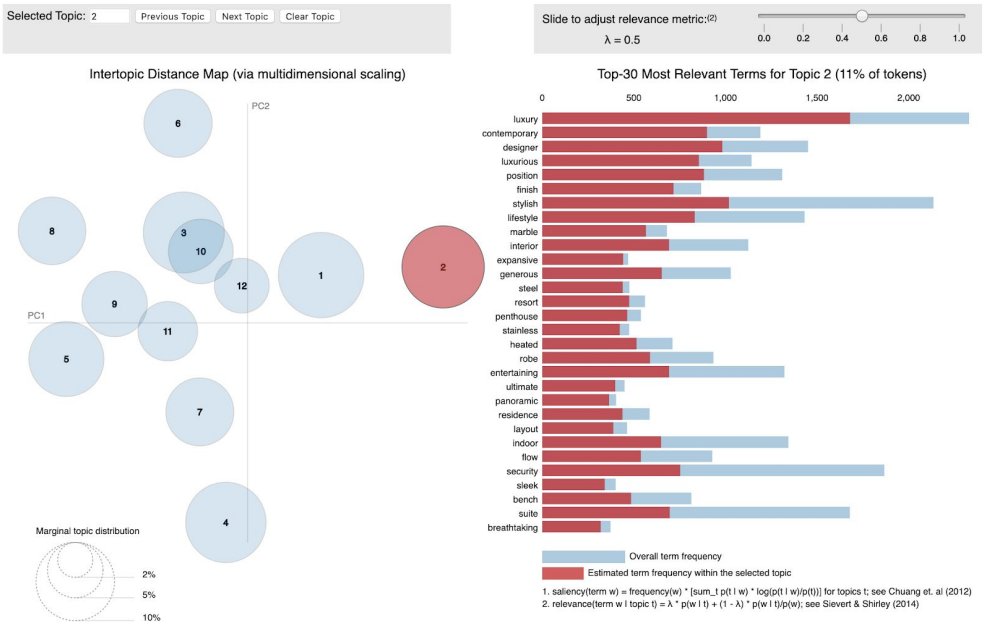
The inertia plot on the left did not show any points with a significant change in the slope of the curve. According to the silhouette plot on the right, k=5 is the global optimum, with Within-Cluster-Sum of Squared Errors (inertia) of 11813 and silhouette score of 0.0165. Typically, we expect to see a global maximum as a peak in the silhouette score curve. The way the curve starts at the maximum and continues to fall until it hits a horizontal asymptote around or below zero indicates that there are no hidden clusters in the word embeddings.

We also performed a manual check of the results. We fit a KMeans with k = 5 clusters, and computed 5 centroids. These centroids should show us any characteristics unique to the cluster, if any. We wrote a script to search through the 1 million word vectors in FastText and return the 10 word vectors that are closest to each of the centroids. Only clusters 3 and 4 showed relevant results, but the words were too generic since there are only five clusters. Results:

```
k=1: ['things', 'stuff', 'brown', 'wooden', 'clothes']
k=2: ['truly', 'classic', 'a', 'stylish', 'charming']
k=3: ['artist', 'expeditions', 'musician', 'adventurers', 'adventuring', 'adventurer', 'journey', 'adventure', 'adven
tures', 'solo']
k=4: ['shops', 'hotel', 'boat', 'downtown', 'school', 'town']
k=5: ['either', 'say', 'way', 'go', 'just']
```

The reason for such a low silhouette score (0.0165) could be that the preprocessing of the listing descriptions wasn't done optimally (too much or too little was removed) to show clusters, or that there are no underlying clusters in the first place. We tried other preprocessing methods, such as normalizing the vectors, removing all adjectives, all nouns, both adjectives and nouns, as well as removing everything except adjectives and nouns, but none of the models produced results in favor of our alternative hypothesis. If we eliminate the possibility of suboptimal preprocessing, the other explanation is the lack of underlying clusters. It could be possible that there are just too many topics discussed in each listing description, that any unique trait is undermined by other generic traits described by uncommon words. One way to address the issues is to create my own word embeddings by training neural networks only on texts from Airbnb website to learn context-specific words. This would prevent words like "adventurers" and "adventuring" being included as separate words in the word embeddings (see k=4 above).

To check the number of topics generally present in listing descriptions, we also performed topic modeling on listing descriptions using Latent Dirichlet allocation, or LDA, and visualized the results. LDA is a statistical approach [8] to identifying unique terms within a specified number of topics in documents, similar to KMeans in that the number of topics must be given. The difference between this approach and the KMeans model used in this project is that the KMeans was trained on the preprocessed average word embeddings, while LDA only looks at vectorized tokens. In other words, unlike the previous approach, LDA simply looks at the term frequency and the inverse document frequency of all the terms with respect to the listing description, instead of attempting to represent listing description as an average word embedding. According to LDA, 12 is the optimal number of topics typically present in listing descriptions.



In the visualization, large circles and minimal overlap indicate optimal topic modeling, with an appropriate number of topics selected. For example, topic 2 describes the style and design of the interior of the listing.

V. Conclusion

The approach of converting preprocessed text to average word embeddings and using KMeans to discover clusters was not a suitable solution for our research and dataset, because listing descriptions discuss too many subtopics. When facing a similar clustering problem with a similar dataset, the approach used in this project may be worth trying only if there are not too many subtopics. Contrasting the poor clustering result with clean topic modeling results, it is likely that the high number of global optima for the number of topics correlate with poor generalization in the average word embeddings. LDA can even be used first to check that there are only a few topics present in the dataset, to determine whether the KMeans approach is viable. If LDA shows that there are many subtopics, the KMeans approach may still produce positive results by training own neural networks to output word embeddings specifically for the dataset. LDA and our main approach are not substitutes, as they are fundamentally different models for different problems - the former for identifying subtopics in text, and latter for identifying clusters of some entity in the data, such as listings.

The main conclusion drawn from this research is that our approach of using KMeans on word embeddings is more suitable for situations where the text discusses a smaller set of topics. For example, the approach would be suitable for a dataset of descriptions of over-the-ear headphones, more than descriptions of speakers and earphones and headphones.

VI. References

1. “Sydney. Adding Data to the Debate.” *Inside Airbnb*, insideairbnb.com/sydney/.

2. Xie, Tyler. “Sydney Airbnb Open Data.” *Kaggle*, 18 Nov. 2019, www.kaggle.com/tylerx/sydney-airbnb-open-data.

3. “Selecting the Number of Clusters with Silhouette Analysis on KMeans Clustering¶.” *Scikit-Learn*, scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html.

4. Mimino666. “Mimino666/Langdetect.” *GitHub*, 5 Mar. 2020, github.com/Mimino666/langdetect.

5. “Inverse Document Frequency.” *The Stanford NLP Group*, nlp.stanford.edu/IR-book/html/htmledition/inverse-document-frequency-1.html.

6. “Word Embeddings.” *TensorFlow*, www.tensorflow.org/tutorials/text/word_embeddings.

7. “English Word Vectors · FastText.” *FastText*, fasttext.cc/docs/en/english-vectors.html.

8. “LDA Topic Modelling.” *Gensim*, radimrehurek.com/gensim/models/ldamodel.html.