Analysis of Word Embedding Techniques for Airbnb Location Classification

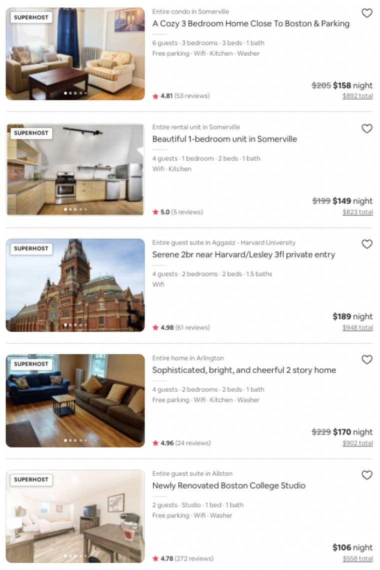
Emily Cheng and Alyssa Unell

May 9, 2022

1 Introduction

Airbnb is a critical development for world travellers and homeowners alike. Founded in 2008, this technological lodging company offers an established platform that provides two main solutions: those who are abroad can find a homey accommodation that hotels cannot provide, while those with spare residential spaces can supplement their income.

Currently, homeowners on Airbnb set their own title for their property. The first five listings for Cambridge, MA, are as follows:



Entire condo in Somerville
A Cozy 3 Bedroom Home Close To Boston & Parking
6 guests · 3 bedrooms · 3 beds · 1 bath
Free parking · Wifi · Kitchen · Washer
★ 4.81 (53 reviews) ~~$205~~ **$158** night
$892 total

Entire rental unit in Somerville
Beautiful 1-bedroom unit in Somerville
4 guests · 1 bedroom · 2 beds · 1 bath
Wifi · Kitchen
★ 5.0 (5 reviews) ~~$199~~ **$149** night
$823 total

Entire guest suite in Aggasiz - Harvard University
Serene 2br near Harvard/Lesley 3fl private entry
4 guests · 2 bedrooms · 2 beds · 1.5 baths
Wifi
★ 4.98 (61 reviews) **$189** night
$948 total

Entire home in Arlington
Sophisticated, bright, and cheerful 2 story home
4 guests · 2 bedrooms · 2 beds · 1 bath
Free parking · Wifi · Kitchen · Washer
★ 4.96 (24 reviews) ~~$229~~ **$170** night
$902 total

Entire guest suite in Allston
Newly Renovated Boston College Studio
2 guests · Studio · 1 bed · 1 bath
Free parking · Wifi · Washer
★ 4.78 (272 reviews) **$106** night
$558 total

Figure 1: The Top Five Listings for Cambridge, MA

There is currently no definitive standard, but we do spot some common features; for listings, some choose to highlight the number of bedrooms, a parking space, an in-house washer and dryer, or their proximity to public transportation.

As such, our goal is to classify the location that a listing is in. Through pre-existing word embeddings, we have created 2 models, one with GloVe and one with Bag of Words, that determine whether a listing is in Hawaii, Los Angeles, New York City, Rhode Island, or Seattle.

2 Data

The data is taken from a table posted on Kaggle by user Kritik Seth: https://www.kaggle.com/datasets/kritikseth/us-airbnb-open-data. The data originates from multiple datasets found on Inside Airbnb and includes 226,000 listings across the United States. The columns of the datasets that we will utilize are the name of the listing as the values to train on (X) and the location of the listing as the labels (Y).

After eliminating irrelevant data, the next step is to eliminate punctuation and special characters; we also made each letter lower case. In particular, for Bag of Words, to reduce dimensionality and the rank of the embedding matrix, we should eliminate filler words, such as "the" or "in", and we would implement this technique in the future to improve results.

Finally, 75% of the data was labeled as training data and the other 25% of the data was labeled as testing data.

3 Word Embeddings

3.1 GloVe

Global Vectors for Word Representation (GloVe) is a current state-of-the-art word encoding approach. It uses pre-trained weight vectors to capture the semantic similarity between words in datasets. Many other common embedding schemes, such as Word2Vec, don't employ this global similarity and as such miss important semantic information that can be used for effective text classification. It uses a co-occurrence matrix to track the probability that two words occur together in order to more appropriately assign embedding values to individual words. This matrix, as such, will be symmetric as it contains the same vocabulary of words for both dimensions.

3.1.1 Embedding Dimensions

GloVe allows for words to be represented in a vector space where the length of each vector for each word is a hyperparameter that can be assigned by the user. It is intuitive that increasing dimensionality of the vocabulary would allow for more nuanced relationships to form between the words, allowing for higher accuracy in model predictions. However, we see here that for simpler NLP problems such as this, there are limits to the effectiveness of increasing the dimension of the vector. We originally tested our dataset with embeddings of size 100, and we got an accuracy of 90.8%. When increasing the embedding size to 300, we obtain an accuracy of 85.9% while decreasing the embedding size to 50 yields an accuracy of 82.3%. As such, we can see that there is no clear benefit to using higher or lower dimensionality for the GloVe word embedding. Further exploration will have to be performed to assess under what conditions these different values are optimal.

3.1.2 Nearest Similarity

Due to GloVe's pre-trained word embeddings, we are able to visualize and assess the similarity of words in our corpus. In our example, we analyze the embedding distances in vector space before training. It would be an interesting area of further research to reassess the distances after training to see how words have shifted from global to local context, as we allow GloVe's global weights to change in response to their usage in our training sentences. We can examine word's nearest neighbors by their Euclidean distance in the vector space. For words that were already in the vocabulary corpus, the nearest neighbors are extremely logical. For example, the word "luxury" is closest to these 10 words: 'luxury', 'luxurious', 'upscale', 'hotels', 'hotel', 'boutique', 'cars', 'condos', 'rental', 'premium'. These associations make intuitive sense to us and confirm that our GloVe embeddings are operating as expected. We can then look at words that are new to the GloVe embedding dictionary, such as 'bdrm' which appears frequently in airbnb advertisements (an abbreviation for 'bedroom') but is not pre-defined in the GloVe embeddings. As such, during this analysis, we set all 'missed' words vectors pre-training to a vector of size (embedding size) filled with zeros. The words that 'bdrm' are closest to are: ", '[UNK]', 'bdrm', 'dtla', 'bedstuy', 'kamaole', 'brba', 'bedbath', 'ilikai', 'bdba'. Associations here such as 'bdba' make sense as further abbreviations recognizing home features, and are most likely used in similar contexts as the original 'bdrm'.

We can then map out these embeddings in a feature space to observe similarities or patterns between semantically similar words. We use T-distributed Stochastic Neighbor Embeddings here to visualize our vectors in a 2 dimensional space. As such, we must note that many aspects of the dimensionality are not preserved and complex relationships between the words cannot be viewed from the attached plot, but even with this caveat, we see some interesting patterns arise here. For example, 'Kona', 'Beach', and 'Pool' are all close to one another in the space. 'Penthouse' and 'Townhouse' also are close neighbors.

3.1.3 Trainable: True vs False

While setting the Trainable parameter to be true may cause overfitting in many circumstances, because our dataset includes so many unconventional words particular to Airbnb (such as proper nouns, misspelled words, abbreviations), it is effective for our models weights to be trainable such that words that were classified as "misses" in the embedding (and thus had no embedding going into

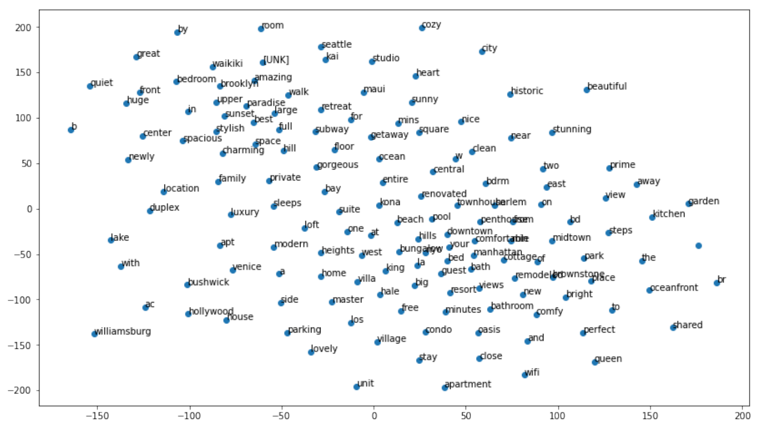


Figure 2: Embedding in Feature Space for GloVe Using TSNE

training) may derive values to aid in classification as more examples of this special word are seen by the model.

Our accuracy for when the trainable parameter was true with the embedding expressed in 100 dimensions was 90.81%. When the trainable parameter was false, the accuracy was 82.27%. We relate this difference to the fact that the text corpus as generated from the Airbnb captions contained 7649 words that were recognized by GloVe and hence given appropriate embeddings, and 8271 words that were embedded equally as vectors of 0. As such, over half of the data presented to the model to be trained on was irrelevant to the weights of the model, leading to a lower accuracy.

3.2 Bag of Words

Bag of Words (BOW) is a popular embedding technique that we used to determine what baseline accuracy for the model could look like with a very simple word embedding scheme. BOW transforms training and testing sentences into vectors where each word has a distinct value that is unrelated to other words. We handle minor word context with introducing bigrams into our BOW encodings (words that commonly occur together such as New and York would have one BOW value for when they are present in a sentence as New York), but overall, much semantic information is lost in this process. Additionally, our methodology included all adjacent pairs of words in sentences as bigrams, regardless of their frequency in the overall dataset. While this did not seem to have a significant impact on our results for this experiment, it is expected that as a vocabulary corpus grows, pre-processing will have to be implemented to ensure that the BOW dimensionality is regularized. Furthermore, we did not remove stop words from the text, and as such, our matrix for BOW was sparse and high dimensional. In more complex classification problems, this could serve as a significant problem as well as a computational bottleneck. However, we saw relatively high accuracy with the BOW model.

When we ran the BOW model with the original dataset, we received 84% accuracy. However, as we can see in the figure, the class '3' has fairly low accuracy in comparison to the other classes. This is most likely due to an imbalance of the dataset. An implicit bias is introduced into the model since every test sentence is statistically less likely to be from class '3' (which is Rhode Island). As such, the model sacrifices accuracy of this class for a higher overall accuracy on the classification task in which it is fit to.

When we rebalance the dataset so that all classes are equally represented, we expect the overall accuracy to decrease since we are training on less examples, but the accuracy for class '3' and other underrepresented classes to increase. This is exactly the result that we saw– the accuracy dropped to 79% but the Area under precision recall curve (AUPRC) increased from 63% to 86% for the third class.

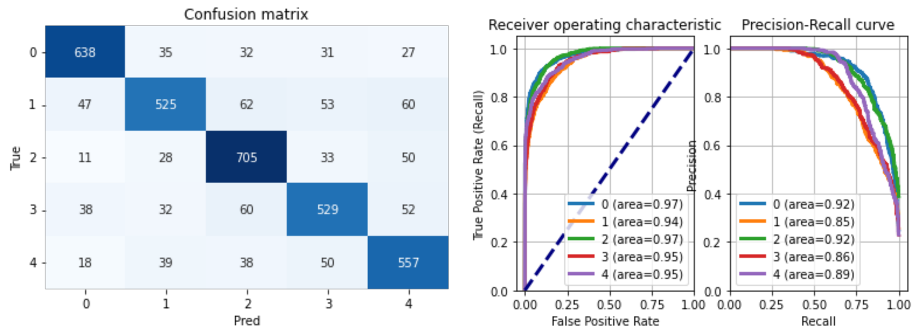


Figure 3: Confusion Matrix and AUPRC for Balanced BOW Embedding Model

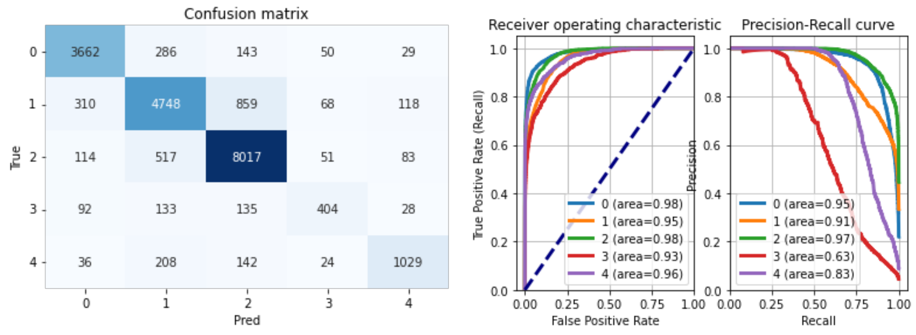


Figure 4: Confusion Matrix and AUPRC for Unbalanced BOW Embedding Model

3.3 Visualization

For visualizing the word embeddings, we utilized the web application Tensorboard, Tensorflow's toolkit. We take advantage of its feature to project embeddings into lower dimensional spaces, so that we can view and understand relations between words in a 3-dimensional space. For more context, we first import the projector plugin from Tensorboard and create weights from the embedding layer to be trained. Then, we visualize the embeddings in a 3d graph from the model we created.

The dimensionality reduction technique we chose is Principal Component Analysis (PCA), while UMAP and t-SNE are offered as other options. We also toggled between using cosine similarity or euclidean distance for highlighting a specific word's closest neighbors.

PCA is derived from Singular Vector Decomposition (SVD), which is arguably the most important concept in Linear Algebra. In essence, the singular value decomposition for the matrix A is

$$A = \sum_i \sigma_i v_i v_i^t.$$

In contrast to eigenvalue decomposition, SVD can handle sparse, yet gigantic, matrices frequent in the field of natural language processing. Generally speaking, PCA projects the data onto the hyperplane closest to the data in a manner that preserves the key information, or the most variation, in the dataset. By choosing the right principal component, or hyperplane, we can capture the maximum amount of information with as few dimensions as possible.

On the other hand, UMAP offers some advantages of note:

1. exponential probability distribution in high dimensions
2. absence of normalization, which reduces computing time
3. nearest neighbors
4. binary cross-entry as cost function, which is implemented in gradient descent and preserves global structure of the data
5. stochastic gradient descent, which allows for faster computations and less memory consumption

Moreover, t-SNE is now an outdated model usurped by the strengths of UMAP. t-SNE fails to scale rapidly with larger sample sizes and requires excessive memory to accomplish analysis beyond simple clustering. Ultimately, t-SNE is a pure Machine Learning algorithm, whereas UMAP is based off sound mathematical principles.

PCA is the appropriate choice because at the top level, it's an efficient eigenvalue method that preserves key information while reducing the dimensions of a dataset. In terms of linear algebra, recall matrix A can be diagonalized such that the direction of the vectors u_i in $A = \sum_i \sigma_i v_i v_i^t$ represent the ith principal, or "most important," direction. SVD extracts data in the most important directions, or in the direction that contains the highest variances in our data set. Hence, we find linearly uncorrelated orthogonal axes, aka the principal components. As such, we can project the high dimensional data into a small m dimensional space.

Below is the tensorboard depiction of the GloVe version of our model:

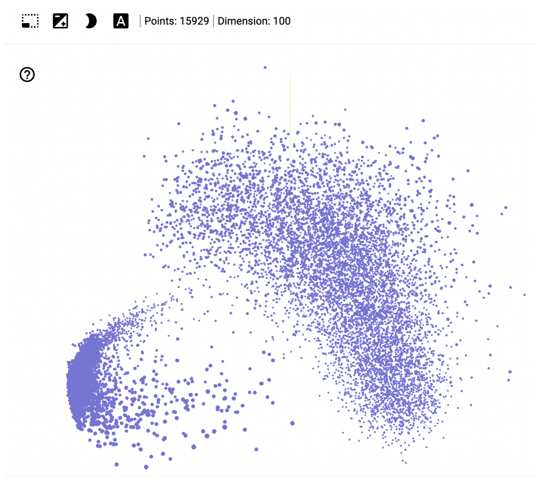


Figure 5: Tensorboard Embedding Visualization of GloVe

We immediately spot 2 clusters, yet upon further examination, we failed to detect any similarities for words in the entire cluster. Upon examining some specific words, we can see some understandably correlated words. According to our model, the words most closely related to "sand" are "firepit," "halls," and "wailua." Additionally, some other words include "surfs, "hottub," and "patio." We attribute this to the fact that GloVe, in the trainable setting, only saw some of these niche, vacation-related words a few times and these words did not come with the pre-existing embeddings. Therefore, the NLP model was unable to develop a proper understanding between Airbnb listing related words. We can explore increasing the learning rate for the trainable embedding weights so that these few examples are better equipped to create meaningful embedding values.

4 Conclusion

We are fascinated by the potential of machine learning and its improvements made possible by a deep understanding of linear algebra. In this project, we explored its relevance to Natural Language Processing (NLP), and more specifically, the mechanics of two types of word embeddings: GloVe and Bag of Words. We compared and contrasted the embedding types and explored the mechanics of dimension reduction in data (embedding) visualization. While both methodologies are widely used in practice today, GloVe with the trainable parameter set as true is the superior choice for classifying Airbnb listing locations.

References

[1] https://nlp.stanford.edu/projects/glove/

[2] https://becominghuman.ai/mathematical-introduction-to-glove-word-embedding-60f24154e54c

[3] https://cs.nyu.edu/ fergus/teaching/vision_2012/9_BoW.pdf

[4] https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17

[5] https://setosa.io/ev/principal-component-analysis/

[6] https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29

[7] https://www.tensorflow.org/tensorboard

[8] https://analyticsindiamag.com/hands-on-guide-to-word-embeddings-using-glove/

[9] https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794

[10] https://keras.io/examples/nlp/pretrained_word_embeddings/