Beer2Vec

Hyunyung Boo

March 2018

1 Definition

Project Overview

The world of beer is expansive and the huge number of options can be intimidating for weekend Bud-drinkers and craft beer enthusiasts alike. Brews can range from a light lager to a creamy, nutty porter, and mouth-puckering sours challenge even the most seasoned of beer drinkers. For those seeking to find their next favorite drink or simply to expand their palettes some assistance navigating the universe of beers would be welcome. One tool can be found at http://www.recommend.beer/.

Problem Statement

The challenge is to build a recommendation engine that can recommend a beer that matches the user's tastes but may be new and undiscovered. The performance of such a model would be measured by using user reviews from a testing dataset. Each user has a set of beers which were rated positively and when given a subset of these beers as an input, the model output would be expected to include beers in the remaining set. This is a type of classification problem which takes input characteristics such as bitterness, sweetness, and alcohol content and assigns each beer to one of two labels, recommended or not recommended. A tutorial for implementing such a system based on collaborative filtering can be found at http://blog.yhat.com/posts/recommender-system-in-r.html.

Project Design

The desired solution to this problem is for the algorithm to return for a particular user a list of 10 beers which were not included in the user's history of preferences but which has been classified as similar enough to the input to be recommended.

One way to build a recommendation system is content-based filtering, which uses the metadata around items to calculate similarity and thus make recommendations about new items[1]. Collaborative filtering on the other hand is based on a pool of past user actions, such as a

previous reviewer's history of preferences [1]. The downsides of collaborative filtering is that it requires user history, and may not perform well for users with uncommon preferences.

On the other hand, collaborative filtering can be used for datasets with little metadata and no domain knowledge of the items being recommended. A whiskey recommender was built using similar principles [2] using data scraped from reddit and applying the Word2Vec algorithm.

Word2Vec is an algorithm which was initially developed to create word embeddings to represent words from a large text input. The embedding captures semantics of words or ideas based on their proximity to other words or items in the input. After generating a vector space of beers based on reviews from beeradvocate.com, the space can then be used to find beers which are similar to a user's known preference based on hidden parameters that the algorithm calculates.

Each reviewer's history of reviews will be curated to select positive reviews (4+). These histories will be used as the input to an algorithm called Word2Vec which creates a vector space based on inferred features of each beer. To use the model generated by Word2Vec, user vectors will be generated based on an example user's list of highly rated beers. The 10 beers with the highest cosine similarity to the user's vector will be recommended.

Metrics

The final model will be evaluated by curating positive (¿= 4.0) reviews from each user in the test dataset.

```
{userId: 'testSubject1',
 positive_reviews: {
        'Dark | AB': 5.0,
        'Light | Coastal Creamery': 4.5,
        'Sour | Nice Place': 4.9
        ...

 }}
```

Given a test user above, 10 percent from the user's preferred list is withheld to use as the testing dataset. The remaining beers will be input into the Word2Vec based algorithm and the success of the model will be evaluated based on how accurately its recommendations match the test dataset, that is what percentage of the 10 recommended beers are contained in the training set. The results are limited to 10 because this system will be used to recommend beers to actual users and more than 10 suggestions would be difficult for a user to process.

2 Analysis

Data Exploration

A public repository of beer reviews exists at www.beeradvocate.com. Users are able to create a profile and rate any beers in the beeradvocate database on a scale from 0 to 5. The reviews from the 250 most prolific reviewers were collected to represent various flavor profiles and tastes. The features of interest are 1) a unique user identifier e.g. userId 2) unique beer identifier e.g. name and brewery 3) the numeric score between 0-5 given each beer by the user. This list of beers for each reviewer will represent their history of preferred beers, or taste profile.

Past analysis on beers had been done by others on a beeradvocate repository at http://snap.stanford.edu/data/web-BeerAdvocate.html but it has since been removed. For the purpose of this project, data was scraped directly from the website. An example of the data gathered from a single user's activity is shown below.

```
{ userId: 'morebeer',
  reviews: {
        'Tasty Caramel | Best Brewery' : 4.7,
        'Ok Beer | Bob's Beers': 3.3,
        'Salty Stout | Mountain Hut': 4.1
  }
}
```

I am defining a 'prolific reviewer' as a user who has submitted 1000 - 10,000 distinct reviews. By scraping the history of 250 of these users, roughly 1,000,000 data points will be collected with approximately 60,000 unique beers represented in the model.

The web scraping script was implemented in python using BeautifulSoup which provides an easy to use HTML document element tree for parsing. The relevant information from each review was stored as a json object and written to a text data file.

Exploratory Visualization

One interesting thing about data from beeradvocate is that popularity or the number of reviews is not correlated with high numeric ratings. The top 250 most active reviewers have rated highly some beers which have few ratings overall. From the following chart, beers with a rating of 4.0 have the most reviews, with long tails on either end.

The tails are evidenced by a high variance in the data. The mean is 3.79 which is close to the highest peak in Figure 1 with a rating of 4.00.

Average	Var	StD
3.79	266	16.33

Table 1: Statistics of scraped data

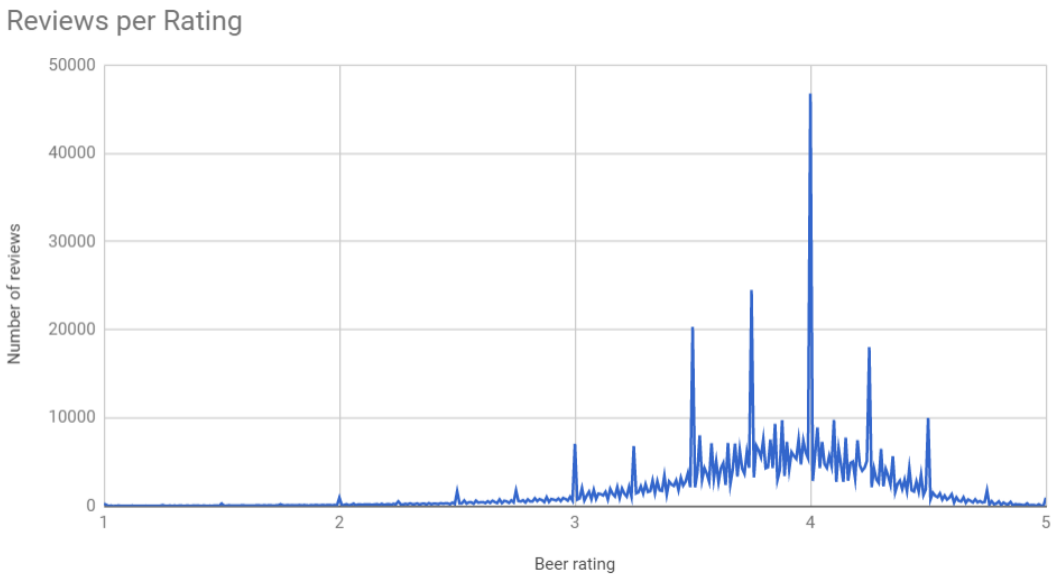

Figure 1: Chart of frequency per numeric rating.

Algorithms and Techniques

Gensim Word2Vec model

Word2Vec is a model developed by Google for use in the NLP space. It is well suited to recommendation systems because the model is capable of learning semantics about different "words", or beers in this case, from contextual clues. In the example of the English language, the space produced by Word2Vec from an input of various sentences is able to express analogies such as the following:

```
king - queen = man - woman
```

The above is not entirely accurate, because the algorithm represents each of the words 'king', 'queen', 'man', etc as vectors in the word space. An operation on these vectors such as

```
king - queen - man
```

should produce a vector which is very close to the vector for the word queen. The metric for closeness in the Word2Vec vector space is cosine similarity, defined as

$cos(u, v) = \frac{u*v}{||u||||v||}$.

Two vectors which represent similar words or concepts would be expected to have high cosine similarity [6]. Because it is able to learn these implicit features about input words, it can also infer features about different beers which are similar without needing to parse text of beer descriptions or chemical composition. Gensim provides a python implementation of Word2Vec which was used for this project [7].

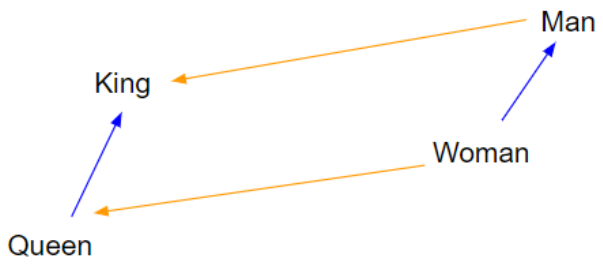

Figure 2: Word2Vec vectors example

While other representations of word vectors exist, the advantage of the Word2Vec implementation is that it is both efficient and preserves the context of words. The trick is in representing a word based on neighboring words or word 'co-occurence' in sentences or phrases. Without this, each word will need to be one-hot encoded in a vector the size of the entire vocabulary. For the English language, this is close to n=200,000 words according to the Oxford dictionary [8] meaning that each input vector will be size n. For Word2Vec, the size of input vectors is many orders of magnitude smaller depending on the size of the sliding window used.

3 Methodology

Data Preprocessing

Data was scraped from https://beeradvocate.com using the parser.py implementation. There were no missing values in the top 250 reviewers used. To create a string "word" for the Word2Vec model input the beer name was joined with colons to the brewery name and beer style. An example of one word constructed from the following scraped data is as follows:

```
Beer: Le Roar Grrz Beery
Brewery: Bullfrog Brewery
Style: American Wild Ale

Word: Le_Roar_Grrrz_Berry:Bullfrog_Brewery:American_Wild_Ale
```

For the purpose of this analysis the numeric rating given by a user was only considered to categorize the review as positive or negative. The threshold used was 4.0 inclusive. An example of a complete "sentence" of 10 words is as follows.

```
Spirit_Animal:Right_Proper_Brewing_Company_-_Shaw_Brewpub:Saison
 _/_Farmhouse_Ale L.A._Gold_West_Coast_Keller_(Collaboration_With
```

```
 _Highland_Park_Brewery):El_Segundo_Brewing_Company:Kellerbier
 _/_Zwickelbier Gadabout_With_Centri_Coffee:Topa_Topa_Brewing:Oatmeal_Stout
Foggier_Window:Monkish_Brewing_Co.:American_Double_/_Imperial_IPA
Berlinerish_Blueberry:Cellador_Ales:Berliner_Weissbier
Bourbon_Barrel_Aged_Fayston_Maple_Imperial_Stout:Lawson's_
Finest_Liquids:American_Double_/_Imperial_Stout Irish_I_Was_A_Little_Bit
_Taller:Noble_Ale_Works:Irish_Red_Ale Lips_Of_Faith_-_Super_India
_Pale_Ale_(Alpine_Collaboration):New_Belgium_Brewing:American_Double_/_
Imperial_IPA Grey_Monday:The_Bruery:American_Double_/_Imperial_Stout Contraband:
Black_Market_Brewing_Co.:California_Common_/_Steam_Beer
```

Implementation

The beer name, brewery, and style were concatenated into a single 'word' to use to train the model by the following method.

```
class FileToSentence():
  def __init__(self, filename):
    self.filename = filename

  def __iter__(self):
    for line in open(self.filename, 'r'):
      ll = [i for i in unicode(line, 'utf-8').split()]
      yield ll
```

The gensim implementation of Word2Vec was used to build and train the final model. A file containing one user sentence per line was used as the training data input. The Word2Vec documentation can be found at https://radimrehurek.com/gensim/models/word2vec.html.

This implementation allows a trained model to be saved after it is initialized and trained. Once a model was produced, it was reloaded from disk to use for evaluation or generating recommendations without having to retrain.

Refinement

The parameters which were important to tune for this model include window and min count. Window size determines the maximum distance between the given and predicted word in a sentence. This is a parameter which is important with data involving real English sentences because the meaning of a phrase would change when two words are adjacent to each other (window=1) versus two words apart (window=2). For predicting preferences of beers, proximity of beer reviews would indicate that the reviews were written in the same window of time. For

example, whether beer A and beer B were consecutive positive reviews or whether they were reviewed months apart may indicate something about a user's changing preference.

Min count determines how many times a word must occur before being used to train the model. Because of the number of beers which are sparsely recommended, using a low number for the min count would improve our results. Table 2 shows the model results given different values for these two parameters.

A third parameter important for Word2Vec is the dimension of the resulting model space. The following table shows that for a window of 1 and min count of 1, the highest total positive results are found by using a dimension of 50. This makes sense because each dimension in the vector embedding space represents a possible feature and enough dimensions are required to fully represent the embedded vectors. However, using a dimension which is excessively large may run the risk of overfitting which may have happened for a dimension of 100.

window	$\min_count$	results	pos	neg	avg
1000	1	49	26	23	1.14285714286
1000	2	30	14	16	0.766666666667
1000	3	21	9	12	0.761904761905
1000	4	16	12	4	1.75
1000	5	13	9	4	1.38461538462
500	1	49	20	29	0.897959183673
500	2	30	15	15	0.9
500	3	21	9	12	0.809523809524
500	4	16	7	9	0.9375
500	5	13	8	5	1.15384615385
100	1	49	38	11	1.65306122449
100	2	30	20	10	0.9
100	3	21	15	6	1.42857142857
100	4	16	9	7	0.875
100	5	13	6	7	0.769230769231
50	1	49	42	7	2.38775510204
50	2	30	22	8	1.5
50	3	21	10	11	0.809523809524
50	4	16	9	7	1.25
50	5	13	8	5	1.15384615385
10	1	49	44	5	2.81632653061
10	2	30	27	3	2.2
10	3	21	14	7	1.57142857143
10	4	16	14	2	1.75
10	5	13	10	3	1.46153846154
1	1	49	45	4	2.55102040816
1	2	30	26	4	2.8
1	3	21	20	1	3.14285714286
1	4	16	14	2	3.125
1	5	13	10	3	3.0

Table 2: Window size, min count

window	min count	dimensions	results	pos	neg	avg
1	1	5	49	36	13	1.75510204082
1	1	10	49	43	6	2.69387755102
1	1	50	49	45	4	2.61224489796
1	1	100	49	44	5	2.42857142857

Table 3: Dimensions

As shown in Appendix 1, the three parameters were fitted to maximize the total number of viable results and the positive rate using grid search. The number of total results, or successful queries decreases with increasing min count. The number of positive results generally increases with decreasing window size. Generally, a dimension of 50 resulted in the best positive rate.

4 Results

Model Evaluation and Validation

A challenge of using a data source like beeradvocate.com is that the reviewers tastes tend to skew away from very common drinks such as Budweiser or Coors. Budweiser has an average rating of 2.36, with 242 out of 6,333 users rating it a 4.0 or higher. Also, different datasets may contain different beers which makes it difficult to validate algorithms across different sources. This model may not well serve someone who prefers mainstream beers that were not well rated on this website. This approach will also be less useful to newcomers to beer who have not built up a taste preference as the algorithm depends on a history of known preferences.

The limits of the dataset chosen for training and the dataset of the benchmark model make it difficult to properly validate these recommendation models. The Word2Vec model recommended beers with 89.8% agreement in the test dataset. However, the same dataset validated on the benchmark model had a positive rate of 3% suggesting that the benchmark model may have had different or smaller database of beers. To get around this limitation, a more preferred validation would be to use real users as a test by asking them to rate 10 random beers versus 10 which were recommended by the model. A strength of Word2Vec is the ability to interpret contextual clues and information, and a weakness is that it requires ample amounts of user history. This makes it ideal to use in a setting such as a website like beeradvocate.com, or another application (such as Untappd) which already has many active users and histories and an expansive database of beers.

TSNE visualization

A problem that came up while trying to validate the model is the sheer number of unique beers which exist in the dataset. No one user has reviewed each of the hundreds of thousands of beers which exist on beeradvocate.com. This proved a challenge when applying my validation method since often the test dataset contained beers which the model had not encountered before.

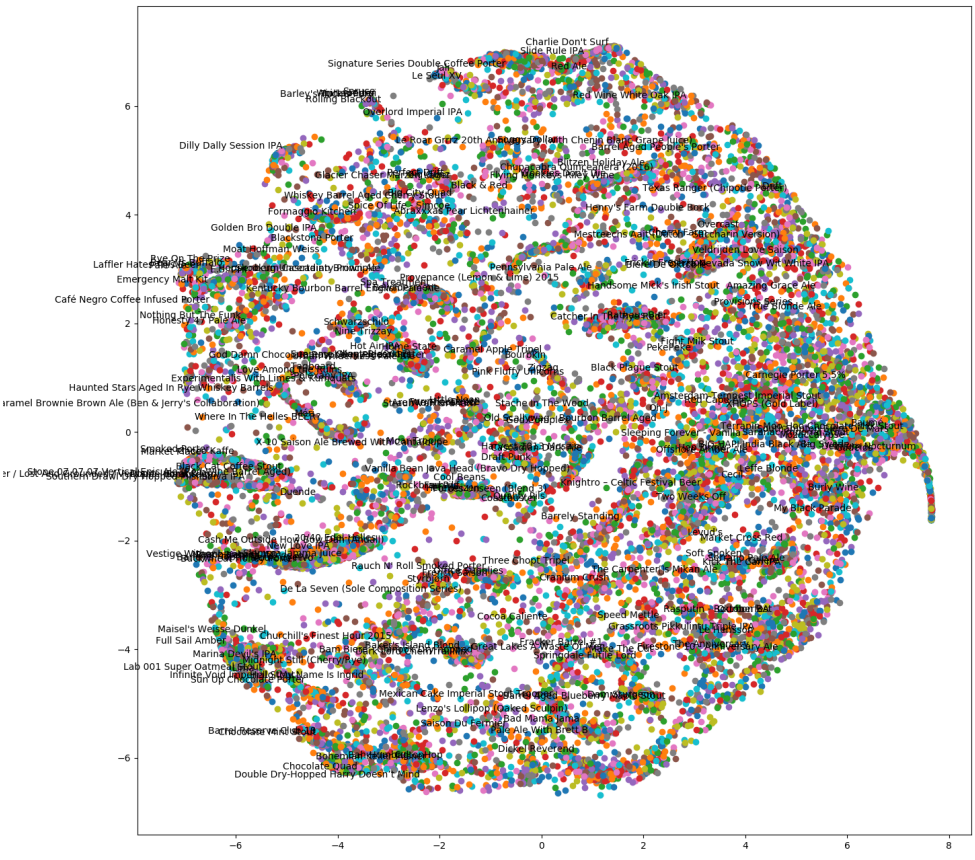

Figure 3: TSNE visualization of trained beer2vec model.

Justification

The model performed best when trained with $windowsize = 1$, $mincount = 1$, and $dimension = 50$ as shown in Table 2 and Table 3. Visualizing the final vector space produced by word2vec is a challenge due to the high number of dimensions in the resulting space. The data needs to be mapped to a 2 dimensional space while preserving the relationships between data points and the features that the space represents. An algorithm for dimensionality reduction for this purpose is called t-distributed stochastic neighbor embedding (5). It can be used to transform the complex output of a neural network to a lesser-dimensional space which can be drawn as a scatter plot (Figure 3). Since related vectors are represented as nearby points the idea of clusters of points having similar features is preserved in the output of TSNE.

Figure 4 focuses on a cluster of the TSNE plot which stands out for representing many porters and stouts which are similar dark beers. Mapping the sampled beer labels to the style shows the following categories represented in this cluster:

```
Milk/Sweet Stout
Irish Dry Stout
American Wild Ale
Oatmeal Stout
American Blonde Ale
Russian Imperial Stout
Irish Red Ale
American Double/Imperial Stout
Baltic Porter
American Stout
Sweet Stout
American Double/Imperial IPA
Dark Ale
Saison/Farmhouse Ale
English Barleywine
Belgian Pale Ale
Flanders Red Ale
```

This cluster would represent a preference of someone who enjoys mainly stouts and porters. Both of these beers and barleywine are generally stronger drinks than other types of beers such as lagers and pale ales. Based on this model produced by Word2Vec, an individual who prefers stronger or darker beers may have luck sampling a new brew from this cluster.

Benchmark Model

A similar model has been built and is available at https://www.recommend.beer. The method used by this recommendation system is based on creating a collaborative filtering matrix. The

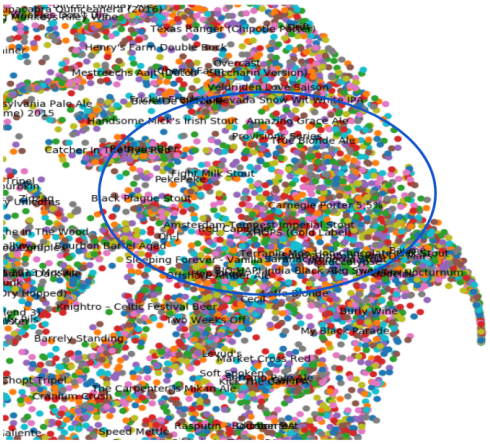

Figure 4: TSNE visualization of trained beer2vec model.

evaluation metric above can be used to compare the results of this model to the model proposed here by using the same testing datasets as inputs to the website recommender and evaluating the accuracy of its prediction.

Using the benchmark model at recommend.beer poses similar problems to the word2vec model in that many beers in the testing dataset were not represented in the model. The sparsity of some reviews makes it difficult to build a comprehensive model.

Out of 250 test users, 34% did not have enough beers found in the model to generate a prediction. 65.2% did not have matches, while 2 users came up with a match resulting in a positive rate of 3% in the beers which were found in the database. Although there were very few exact matches in the generated recommendation, the recommender often suggested beers which were very similar to others contained in the test data. For example, user 3 had a sour beer (Miss Beehavior) in their history, and two different sours (Panil Barriquée, Oud Beersel Oude Gueze) were suggested. Similar recommendations were noticed where the user had several barrel-aged beers in their history and had different barrel-aged ales recommended.

5 Conclusion

Freeform Visualization

Table 4 contains several examples of cosine distance between beers in the dataset. In the first row, Chocolate Stout and Le Temps Noir (which is an Imperial Stout) have a high cosine similarity. This may be expected because both beers are of a similar style and flavor profile. In contrast, the second row shows that Chocolate Stout and Nelsin Sauvin (a Farmhouse Ale) have a lower cosine distance presumably because they are dissimilar. However, one of the key advantages of using Word2Vec in a recommendation system is that entities are placed not based on explicit features such as color, flavor, and style, but based on co-occurence in a sentence.

This means that Chocolate Stout and Le Temps Noir have cosine similarity not because they are both dark beers, but because many users that enjoyed Chocolate Stout also liked Le Temps Noir. This may imply something about an underlying feature such as flavor, but this is not always the case. The highest cosine of 0.96 in Table 4 is between Chocolate Stout and Drie Fonteinen Schaerbeekse Kriek (Kriek is a Belgian cherry beer). This may be surprising since a chocolate stout and a kriek have very different flavour profiles, but their closeness in the vector space indicates that many users who like Chocolate Stout also may like Drie Fonteinen Schaerbeekse Kriek.

Beer 1	Beer 2	Cosine distance
Chocolate Stout	Le Temps Noir	0.80
Chocolate Stout	Nelson Sauvin	0.66
Chocolate Stout	Phoenix Kriek	0.71
Chocolate Stout	Drie Fonteinen Schaerbeekse Kriek	0.96
Phoenix Kriek	Drie Fonteinen Schaerbeekse Kriek	0.75

Table 4: Examples of cosine distance

Reflection

The end goal of this project was to have a working recommendation engine which could suggest to a user something new to try. Beer ratings were scraped from https://beeradvocate.com from the top 250 most prolific reviewers, generating 100 megabytes of data. This data was processed to select reviews which were 4.0 and above and a sentence was constructed for each user based on their history of positive beer reviews. The sentence was used as input in a gensim Word2Vec implementation to train a model. As far as expanding a user's beer horizons, the huge dataset guarantees that the model will be able to recommend beers which are new. The downside of working with the dataset was that it was difficult to validate with the existing users since I had to extrapolate whether they would like or dislike a recommendation based on whether the beer was in their history or not. A better test of the recommendation would be to recommend to real users and gather feedback on the recommendations.

Improvement

A possible improvement would be to further curate the dataset to select a subset of users who were prolific reviewers and reviewed more of the same beers. This would allow for a stronger model to be trained because of more redundant data and validation would be more successful if there was a higher chance that test users had tried the recommended beer or not.

6 Citations

1. Gong, S. (2011). A Personalized Recommendation Algorithm on Integration of Item Semantic Similarity and Item Rating Similarity. Journal of Computers, 6(5). doi:10.4304/jcp.6.5.1047-1054

2. Krzus, M. (n.d.). Retrieved December 04, 2017, from http://wrec.herokuapp.com/methodology

3. Z. (n.d.). Zackthoutt/wine-deep-learning. Retrieved December 04, 2017, from https://github.com/zackthoutt/wine-deep-learning

4. Itoku, E. (n.d.). What is your new favorite beer? Retrieved December 04, 2017, from http://www.recommend.beer/

5. van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing High-Dimensional Data Using t-SNE" (PDF). Journal of Machine Learning Research. 9: 2579–2605.

6. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

7. Gensim, from https://radimrehurek.com/gensim/models/word2vec.html

8. How many words are there in the english language? from https://en.oxforddictionaries.com/explore/how-many-words-are-there-in-the-english-language

7 Appendix

window	min count	dimensions	results	pos	neg	avg
1000	1	5	49	27	22	1.08163265306
1000	1	10	49	25	24	1.18367346939
1000	1	50	49	18	31	0.836734693878
1000	1	100	49	27	22	0.979591836735
1000	2	5	30	17	13	0.8
1000	2	10	30	14	16	0.766666666667
1000	2	50	30	16	14	0.733333333333
1000	2	100	30	14	16	0.666666666667
1000	3	5	21	10	11	0.761904761905
1000	3	10	21	12	9	1.14285714286
1000	3	50	21	13	8	1.2380952381
1000	3	100	21	13	8	1.2380952381
1000	4	5	16	10	6	1.5625
1000	4	10	16	10	6	1.4375
1000	4	50	16	12	4	1.6875
1000	4	100	16	10	6	1.25
1000	5	5	13	9	4	1.07692307692
1000	5	10	13	8	5	1.46153846154
1000	5	50	13	8	5	1.30769230769
1000	5	100	13	8	5	1.07692307692
500	1	5	49	22	27	0.714285714286
500	1	10	49	25	24	0.897959183673
500	1	50	49	19	30	0.775510204082
500	1	100	49	24	25	1.10204081633
500	2	5	30	13	17	0.633333333333
500	2	10	30	18	12	0.933333333333
500	2	50	30	13	17	0.733333333333
500	2	100	30	14	16	0.7
500	3	5	21	11	10	0.714285714286
500	3	10	21	10	11	1.0
500	3	50	21	11	10	0.761904761905
500	3	100	21	12	9	0.761904761905
500	4	5	16	8	8	0.9375
500	4	10	16	9	7	1.1875
500	4	50	16	11	5	1.1875
500	4	100	16	7	9	1.125
500	5	5	13	7	6	1.0
500	5	10	13	7	6	1.30769230769
500	5	50	13	7	6	0.923076923077
500	5	100	13	6	7	0.846153846154
100	1	5	49	30	19	1.38775510204
100	1	10	49	37	12	1.38775510204
100	1	50	49	37	12	1.71428571429
100	1	100	49	36	13	1.67346938776
100	2	5	30	18	12	0.966666666667
100	2	10	30	21	9	1.23333333333
100	2	50	30	18	12	1.0
100	2	100	30	16	14	0.9
100	3	5	21	10	11	1.14285714286
100	3	10	21	16	5	1.19047619048
100	3	50	21	12	9	1.0

Table 5: Grid search results

window	min count	dimensions	results	pos	neg	avg
100	3	100	21	9	12	0.761904761905
100	4	5	16	9	7	1.1875
100	4	10	16	9	7	0.8125
100	4	50	16	7	9	0.6875
100	4	100	16	7	9	0.75
100	5	5	13	5	8	0.615384615385
100	5	10	13	6	7	1.0
100	5	50	13	8	5	0.923076923077
100	5	100	13	8	5	1.07692307692
50	1	5	49	32	17	1.18367346939
50	1	10	49	44	5	1.91836734694
50	1	50	49	40	9	2.38775510204
50	1	100	49	47	2	2.24489795918
50	2	5	30	21	9	1.0
50	2	10	30	20	10	1.43333333333
50	2	50	30	20	10	1.4
50	2	100	30	22	8	1.26666666667
50	3	5	21	18	3	1.28571428571
50	3	10	21	13	8	1.09523809524
50	3	50	21	12	9	1.2380952381
50	3	100	21	18	3	1.38095238095
50	4	5	16	10	6	1.125
50	4	10	16	6	10	0.5625
50	4	50	16	11	5	1.0625
50	4	100	16	4	12	0.4375
50	5	5	13	7	6	1.07692307692
50	5	10	13	8	5	1.38461538462
50	5	50	13	8	5	1.30769230769
50	5	100	13	8	5	1.30769230769
10	1	5	49	37	12	2.02040816327
10	1	10	49	40	9	2.69387755102
10	1	50	49	43	6	2.81632653061
10	1	100	49	41	8	2.59183673469
10	2	5	30	26	4	2.0
10	2	10	30	28	2	2.66666666667
10	2	50	30	26	4	2.5
10	2	100	30	26	4	2.5
10	3	5	21	19	2	1.80952380952
10	3	10	21	18	3	2.04761904762
10	3	50	21	17	4	1.95238095238
10	3	100	21	18	3	2.19047619048
10	4	5	16	12	4	1.5625
10	4	10	16	11	5	1.625
10	4	50	16	13	3	1.75
10	4	100	16	12	4	1.625
10	5	5	13	7	6	1.07692307692
10	5	10	13	11	2	2.07692307692
10	5	50	13	12	1	2.0
10	5	100	13	11	2	2.15384615385
1	1	5	49	36	13	1.75510204082
1	1	10	49	43	6	2.69387755102
1	1	50	49	45	4	2.61224489796

Table 6: Grid search results cont.

window	min count	dimensions	results	pos	neg	avg
1	1	100	49	44	5	2.42857142857
1	2	5	30	26	4	2.16666666667
1	2	10	30	27	3	2.8
1	2	50	30	29	1	3.03333333333
1	2	100	30	26	4	2.7
1	3	5	21	17	4	2.19047619048
1	3	10	21	20	1	3.09523809524
1	3	50	21	20	1	3.19047619048
1	3	100	21	20	1	2.95238095238
1	4	5	16	14	2	2.375
1	4	10	16	16	0	3.1875
1	4	50	16	14	2	3.0
1	4	100	16	14	2	3.25
1	5	5	13	11	2	2.69230769231
1	5	10	13	11	2	3.30769230769
1	5	50	13	11	2	2.92307692308
1	5	100	13	10	3	3.15384615385

Table 7: Grid search results cont.