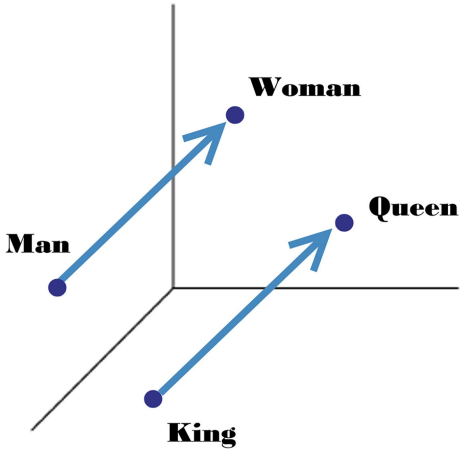
Using word2vec to understand product structure in sales data

NUS CBE
2018.05.23

Introduction

- Objective of word2vec: represent each word as a continuous vector, given a text corpus. Words with similar meaning end up laying close to each other.
- Use vector arithmetics to work with analogies: king - man + woman = queen.
- Numbers are for demonstration.

	word	1	2	3
1	King	1.3	-1.92	0.19
2	Queen	1.2	-0.20	1.64
3	Man	2.4	-1.19	1.64
4	Woman	1.6	0.35	-0.33



Introduction

- Mikolov et al. 2013 uses Google News corpus for training the word vectors. There are 6B tokens and the vocabulary size is 1 million.
- Generalization: this approach can be used when we have a sequence of discrete items.
- Music playlist (Spotify), emoji2vec, prod2vec, medical events (diagnoses, prescriptions, and labs tests).

Use word2vec on sales data

- After removing single-item baskets: 20m tokens (products), vocabulary size (unique products) = 4k, 6m sentences (baskets).

	basketId	productId
1	11	['38057', '37134', '42403', '38800']
2	12	['37677', '54220', '54233', '46086', '54226', '54239', '35624', '46140']
3	16	['36520', '36529']
4	17	['46957', '47863']
5	20	['54804', '54686']
6	31	['43420', '43256']
7	40	['37783', '42722']
8	44	['22448', '4177', '22436', '8552']

Use word2vec on sales data

- productId's word vectors
- how to produce word vector will be explained shortly

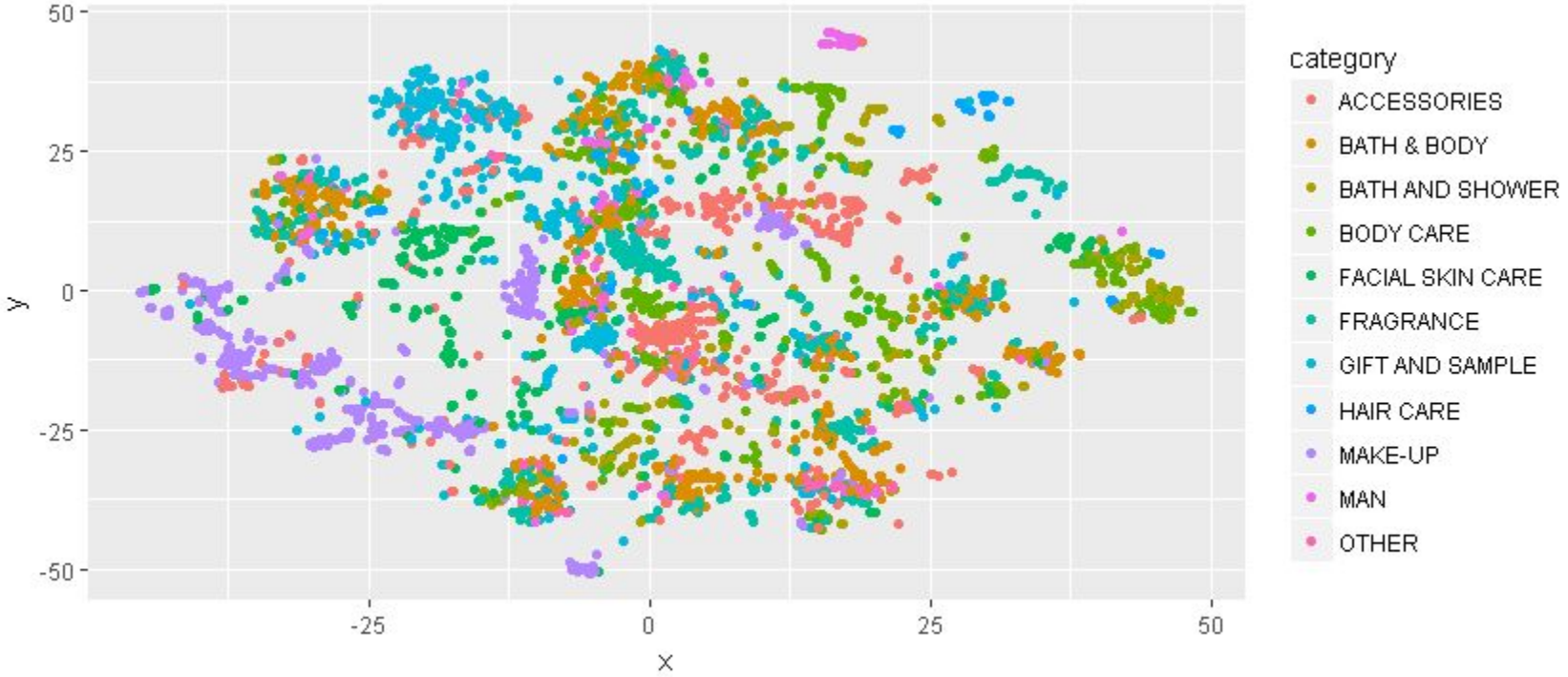
	productId	productName	1	2	3	4	5	6	7	8	9	10
1	2544	NEW ALOE ANTI-PERSPIRANT DEODORANT 50ML	-1.46	1.941	4.57	0.33	-2.37	1.91	3.702	-0.15	-5.01	-2.66
2	2617	NEW VIT.E MOIST CREAM 50ML	1.92	1.497	-0.15	1.70	4.46	-2.30	-5.891	0.55	-1.55	-0.82
3	4887	ALL IN ONE FACE BASE 03	1.49	0.932	0.89	-2.56	1.98	-1.45	-2.905	1.65	2.21	-1.74
4	4896	ALL IN ONE FACE BASE 04	0.38	0.594	0.24	-2.08	1.92	-0.73	-2.275	1.07	1.71	-2.38
5	6040	MMF LIP & CHEEK STAIN 01 PINK ROSE	0.65	-2.355	-1.48	-1.98	-0.97	0.68	-0.096	0.99	1.32	-2.24
6	7996	WHITE MUSK SMOOTH SATIN BODY LOTION 250ML	-3.33	2.493	2.70	-1.17	-2.54	4.93	0.065	2.10	1.72	3.51
7	32768	NEW VIT C F/ CLEAN POLISH 100ML	-1.91	-1.988	5.80	1.08	6.34	1.75	-1.452	-1.88	-0.32	-1.87
8	39357	MORINGA BODY MIST 100ML	-3.03	0.643	-1.34	-0.50	-2.99	1.58	-1.866	1.79	-0.21	0.53
9	39457	VANILLA BODY MIST 100ML	0.68	0.077	-3.13	-2.06	-3.86	1.27	-1.995	0.18	0.38	1.08
10	65160	STAMPCARD 2015	-1.99	-1.793	-2.89	-1.96	-0.98	0.92	1.360	2.08	-0.42	0.30

Product similarity

	productId	56398	58657	27904	27877	2770	8500	65160	42857	36878	4896
1	56398	1	0.578	0.062	0.012	-0.035	-0.316	0.536	0.455	-0.289	0.179
2	58657	0.578	1	-0.02	-0.085	-0.175	-0.109	0.418	0.065	-0.431	0.205
3	27904	0.062	-0.02	1	0.958	0.251	-0.039	0.072	0.223	0.117	0.284
4	27877	0.012	-0.085	0.958	1	0.301	0.042	0.074	0.204	0.103	0.229
5	2770	-0.035	-0.175	0.251	0.301	1	0.188	-0.206	0.037	0.08	0.386
6	8500	-0.316	-0.109	-0.039	0.042	0.188	1	-0.272	-0.306	0.052	0.2
7	65160	0.536	0.418	0.072	0.074	-0.206	-0.272	1	0.042	-0.465	-0.063
8	42857	0.455	0.065	0.223	0.204	0.037	-0.306	0.042	1	0.457	0.311
9	36878	-0.289	-0.431	0.117	0.103	0.08	0.052	-0.465	0.457	1	0.102
10	4896	0.179	0.205	0.284	0.229	0.386	0.2	-0.063	0.311	0.102	1

t-SNE visualization of product vectors

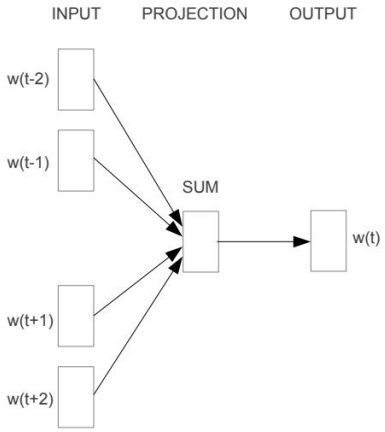
- tsne perplexity =25; vector dimension = 150



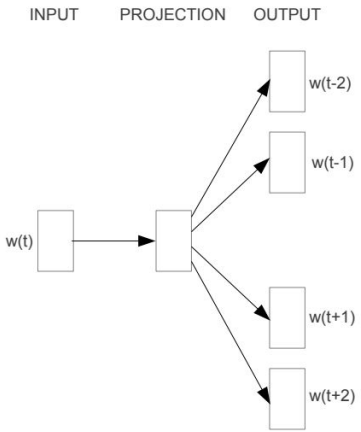
How to get those vectors?

- There are 2 approaches.

1. Predictive (briefly go through):
   - In the original paper by Mikolov
   - Use neural networks
   - Continuous bag of words: input: context words (which surround the target word), output: target word
   - Skip-gram: input: target word, output: context words
   - After the training process, the hidden layer contains the vectors we wanted



CBOW



Skip-gram

How to get those vectors?

2. Count-based:
   - GloVe: Global Vectors for Word Representation (Pennington et al. 2014)
   - Firstly, get the co-occurrence matrix (numbers are for demonstration)

	11788	11814	11823	11849	11860	11870	11907	11921
11788	0	6	0	1	1	0	4	2
11814	6	0	1	0	1	0	3	4
11823	0	1	0	0	0	0	0	0
11849	1	0	0	0	0	0	0	0
11860	1	1	0	0	0	0	0	0
11870	0	0	0	0	0	0	0	1
11907	4	3	0	0	0	0	0	4
11921	2	4	0	0	0	1	4	0

How to get those vectors?

- Once we have the co-occurrence matrix, it becomes an optimization problem:

$$\vec{w}_i^T \vec{w}_j + b_i + b_j = \log X_{ij}.$$

- w*i, wj are the product vectors, Xij is how often product i is bought in the same basket with product j,* bi and bj are scalar bias terms.

How to get those vectors?

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(\vec{w}_i^T \vec{w}_j + b_i + b_j - \log X_{ij} \right)^2$$

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_{\max}} \right)^{\alpha} & \text{if } X_{ij} < x_{\max} \\ 1 & \text{otherwise.} \end{cases}$$

- f(X) helps prevents common word pairs from skewing our objective. For extremely common word pairs (where Xij> xmax) this function simply returns 1. For all other word pairs, we return some weight in the range (0,1)
- 4k words * 150 = 600k parameters to optimize. Often by gradient descent methods.

Notes

- The workflow:
  - transform the data in R
  - then pass to python to use gensim
  - then get back the results and do visualization
- Interpretation of similarity
  - R Shiny

References

- http://www.majumderb.com/assets/documents/prod2vec_initial_report.pdf
- https://omarito.me/word2vec-product-recommendations/