Latent Semantic Analysis and Vector Space Model

Promothesh Chatterjee[*]

[*]Copyright© 2024 by Promothesh Chatterjee. All rights reserved. No part of this note may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author.

Latent Semantic Analysis (LSA)

The basic idea underlying LSA is that text captures knowledge not only through information contained in sentences but also implicitly through co-occurrence of words. These latent or hidden concepts can be extracted and revealed to a certain extent through LSA.

The Need for LSA

Let's look at two sentences:

1) Before winter, we generally see a spike in snow blower sales, the warm forecast will sure reduce sales.

2) The temperature forecast for the coming months is high, this will impact snow removal business.

The two sentences capture similar ideas, yet have only four words in common (the, snow, forecast, will), so the cosine similarity will not be very high. If somehow we could represent "warm forecast" and "temperature forecast high" as a part of a composite dimension, then :

- cosine similarity will improve
- we will be able to reduce the dimensions of the DTM

This is the fundamental problem of synonymy in NLP. Consider another example with two sentences:

1) Banks facilitated sale of arms to Libya

2) With their arms raised, the enemies surrendered at the banks of River Seine.

The words - 'arms' and 'banks' across the two sentences have very different meanings but a cosine similarity measure will not be able to capture the differences. This illustrates the issue of polysemy. About 40% of English words are polysemous. The above two examples illustrate the fundamental issues of synonymy and polysemy in the vector space model. Before we delve any deeper, lets first understand the vector space model and the associated concepts.

The Vector Space Model

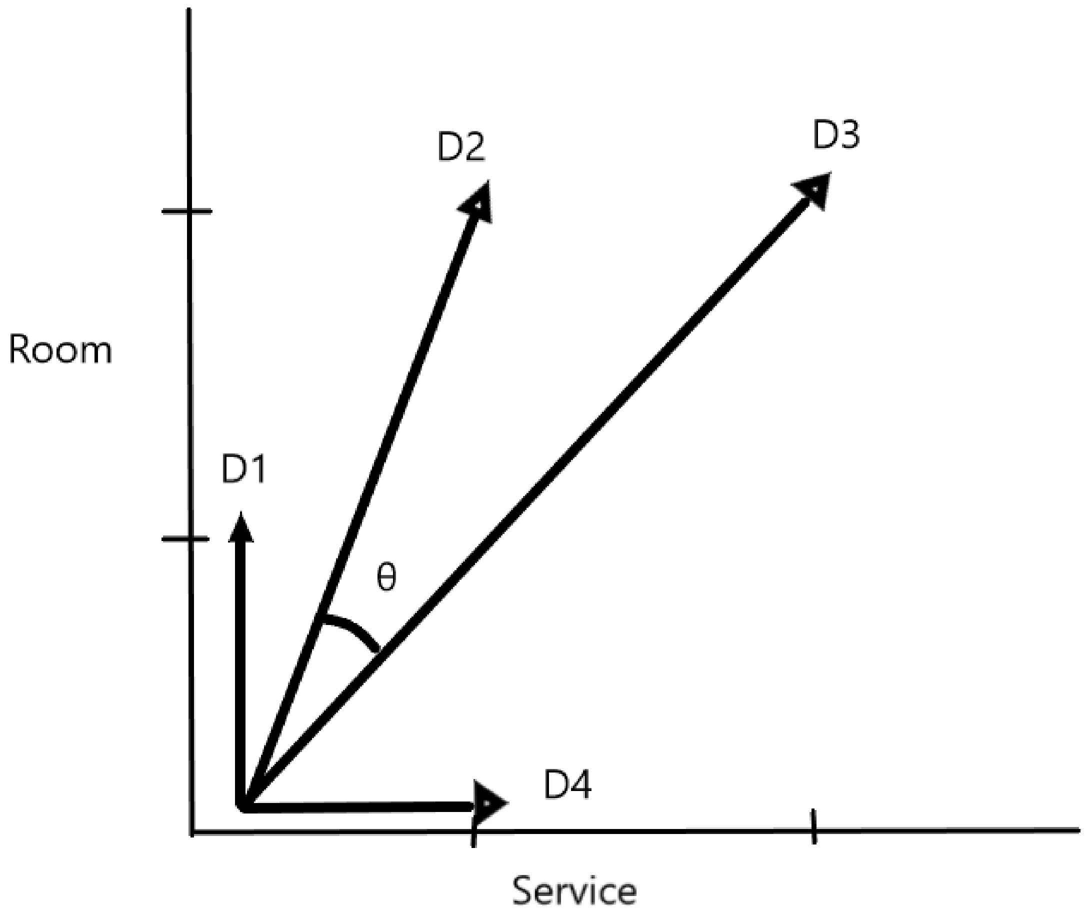
As a simplified example, let's consider two words-'room' and 'service' from our reviews. Say the document-term matrix of these two words and their occurrences across 4 documents is like this:

Docs	Service	Room
D1	0	1
D2	1	2
D3	2	2
D4	1	0

We can also compute the term-document-matrix or TDM which is a transpose of DTM.

Docs	D1	D2	D3	D4
Service	0	1	2	1
Room	1	2	2	0

We can represent the documents in a vector space model. Essentially we plot 'service' on the x-axis and 'room' on the y-axis. Document D1 then has coordinates (0,1), D2-(1,2) and so on. Plotting the coordinates in the vector space we get a figure shown below.



One way to examine distance between the documents is by using Euclidean distance, but Euclidean distance is not a correct measure here. Why?

Say, D1 has coordinates (t11, t12) and D2 has coordinates (t21,t22).

Euclidean Distance between D2 and D1 = $(t21 - t11)^2 + (t22 - t12)^2$

Let's say that in document 2 we simply double the frequency of the terms, so the coordinates become (2xt21,2xt22).Explain what will happen if I do that? The meaning of the document does not change, it simply repeats the terms but the Euclidean distance will increase.That is one reason why euclidean distance, even though it satisfies the the mathematical expectations of distance measure is not used in the context of texts.

If similarity between documents is represented by the angle between the documents, looking at the figure we can easily understand that document D1 is closer to D2 than D3 and so on. This intuitively is the idea behind cosine similarity.

Cosine Similarity

When we represent terms as vectors in space, they have two components 1) direction (where it is going) and 2) magnitude (how big the arrow is). Magnitude is proportional to the number of times the words occur in a document, in another in other words, this is proportional to how big the document is. So the magnitude represents the length of the document and the direction represents how different they are. If D1 and D2 overlapped in the figure instead of going into other direction, we would say that they are pretty similar.

Cosine Similarity (Normalized dot product):

$$sim(X, Y) = cos(\theta_{X,Y}) = \frac{X \cdot Y}{\| X \| * \| Y \|} \equiv \frac{\sum_i^k X_i \times Y_i}{\sqrt{\sum_i^k X_i^2} \times \sqrt{\sum_i^k Y_i^2}}$$

$$-1 \leq cos(\theta_{x,y}) \leq 1$$

Basically, cosine similarity is nothing but dot product (the directional growth of one vector to another) divided by the norm (magnitude) of the vectors. So how do we actually compute it? Let's take a previous term-document matrix. Say we want to compute similarity between document D1, D2 and D1,D3.

Docs	D1	D2	D3	D4
Service	0	1	2	1
Room	1	2	2	0

For computing cosine similarity of D1, D2, let's first compute their dot product.

Dot product D1,D2= $0.1 + 1.2 = 2$

Norm = $\sqrt{(0^2 + 1^2)}.\sqrt{(1^2 + 2^2)}$ =$\sqrt{5}$

Cosine = $2/\sqrt{5} = 0.894$

thus, angle is cosine inverse of $0.894 = 26.6$ degrees

Computing cosine similarity of D1, D3:

Dot product D1,D3= $0.2 + 1.2 = 2$

Norm = $\sqrt{(0^2 + 1^2)}.\sqrt{(2^2 + 2^2)} = 2\sqrt{2}$

Cosine = $2/2\sqrt{2} = 1/\sqrt{2} = 0.707$

thus, angle is cosine inverse of $0.707 = 45$ degrees. Therefore our intuition about D1 and D2 being similar compared to D1 and D3 was correct.

We can extend the idea depicted to here to an n-dimensional space but the concepts remain pretty much the same. Next, we will look at the concepts underlying LSA.