1. ppmi[0][0] = array([1.60893804, 0.   , 0.   ])

$$p(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

   The output ppmi matrix has 0 value, because the input matrix has 0 word counts. Just because an event has never been observed in training data does not mean it cannot occur in test data (data sparsity issue)

2. We can use Laplace smoothing to deal with the above problem. It raises the baseline above 0. Before computing PMI, a small constant k (values of 0.1-3 are common) is added to each of the counts, shrinking (discounting) all the non-zero values. The larger the k, the more the non-zero counts are discounted.

3. Euclidean is length of the most direct line between the two points. Cosine distance takes overall length into account. The similarity part of this (the righthand term of the subtraction) is actually measuring the _angles_ between the two vectors. Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts. We could assume that when a word (e.g. science) occurs more frequent in document 1 than it does in document 2, that document 1 is more related to the topic of science. However, it could also be the case that we are working with documents of uneven lengths (Wikipedia articles for example). Then, science probably occurred more in document 1 just because it was way longer than document 2. Cosine similarity corrects for this. When classifying documents we'd like to categorize them by their overall sentiment, so we use the angular distance. Euclidean distance is susceptible to documents being clustered by their L2-norm (magnitude, in the 2 dimensional case) instead of direction. I.e. vectors with quite different directions would be clustered because their distances from origin are similar.
4. Long Sparse
   Short, Dense - we will use LSA - SVD