CS155 Homework 5

Ty Limpasuvan

15 late hours used; 11 late hours remaining

1 SVD and PCA

1.1 A

SVD: $X = U\Sigma V^T$
$XX^T = (U\Sigma V^T)(E\Sigma V^T)^T$
$= U\Sigma V^T V\Sigma U^T$
$= U\Sigma^2 U^T$
$= U\Lambda U^T$
$XX^T = U\Lambda U^T$
Σ^2 is the eigenvalues of X.

1.2 B

$Av = \lambda u$
We have $v^T Av = v^T(\lambda v) = \lambda v^T v$
$v^T \lambda v \geq 0$
$= \lambda v^T v \geq 0$
If v is diagonal, we have $\lambda \geq 0$
Eigenvalues are analogous to the variation, which cannot be negative.

1.3 C

$Tr(AB) = \Sigma_{i=1}^{N}(AB)_{ii}$
$= \Sigma_{i=1}^{N}\Sigma_{j=1}^{M}A_{ij}B_{ji}$
A matrix and its transpose have the same trace, so we also have $= \Sigma_{j=1}^{M}\Sigma_{i=1}^{N}B_{ji}A_{ij}$
$= \Sigma_{j=1}^{M}(BA)_{jj}$
Which is how we define the trace of BA $Tr(BA)$

1.4 D

We need to store 2N x k + k values to store a truncated SVD with k singular values. We have N points to store and k features of these values. However, an N x N matrix would be represented by the product of matrices of the following dimensions: (N x K)(K x K)(K x N). 2(N x k) + k values are needed; the diagonals of the diagonal matrix need to be stored as well. k values of less than N make storing the truncated SVD more efficient than storing the whole matrix.

1.5 E

1.5.1 i

X has rank N, so its column vectors are described by N bases. This means the eigenvectors also have a basis of N. Thus, we don't need all D points because we only need N bases.

1.5.2 ii

If a matrix isn't square, it does not have an inverse. And if a matrix is orthogonal, its inverse is its transpose. Thus if the matrix isn't square, it can't be orthogonal.

1.5.3 iii

Multiplying the orthogonal matrix with its inverse is equivalent to multiplying the matrix with its inverse, which results in the identity matrix. Its size will be N x N, where N is the number or rows or columns in the original.

1.6 F

1.6.1 i

The psuedo inverse of a matrix is equal to its inverse for an invertible matrix.

2 Matrix Factorization

2.1 A

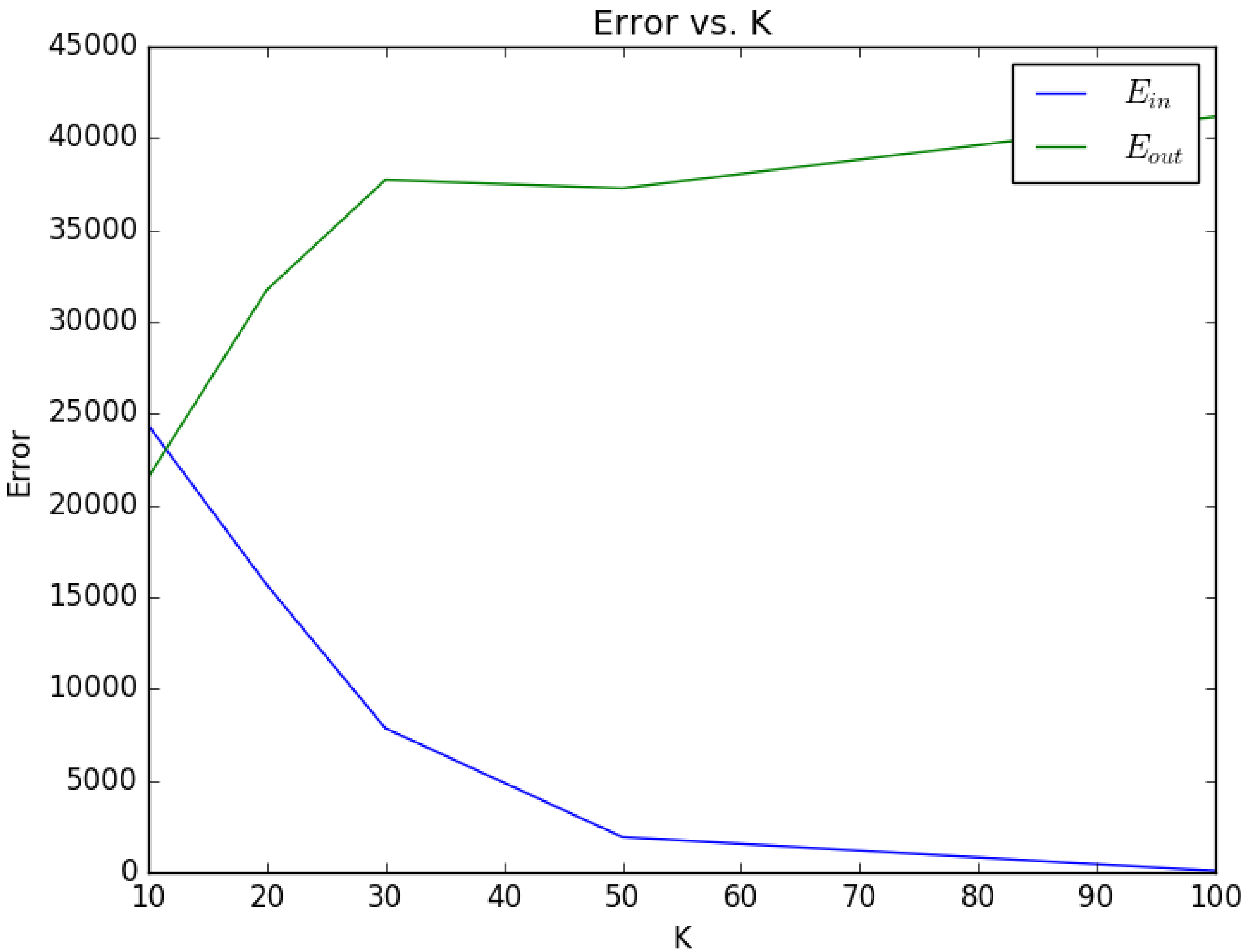
$\partial_{u_i} = \lambda u_i - \sum_j v_j (y_{ij} - u_i^T v_j)^T$
$\partial_{v_j} = \lambda v_j - \sum_i u_i^T (y_{ij} - u_i^T v_j)^T$

2.2 B

To minimize the squared error, we set the gradient equal to zero then follow the steps outlined.
$\lambda u_i - \sum_j v_j y_{ij} + \sum_j v_j u_i^T v_j = 0$
$u_i(\lambda + \sum_j v_j v_j^T) - \sum_j v_j y_{ij} = 0$
$u_i = (\lambda + \sum_j v_j v_j^T)^{-1}(\sum_j v_j y_{ij})$

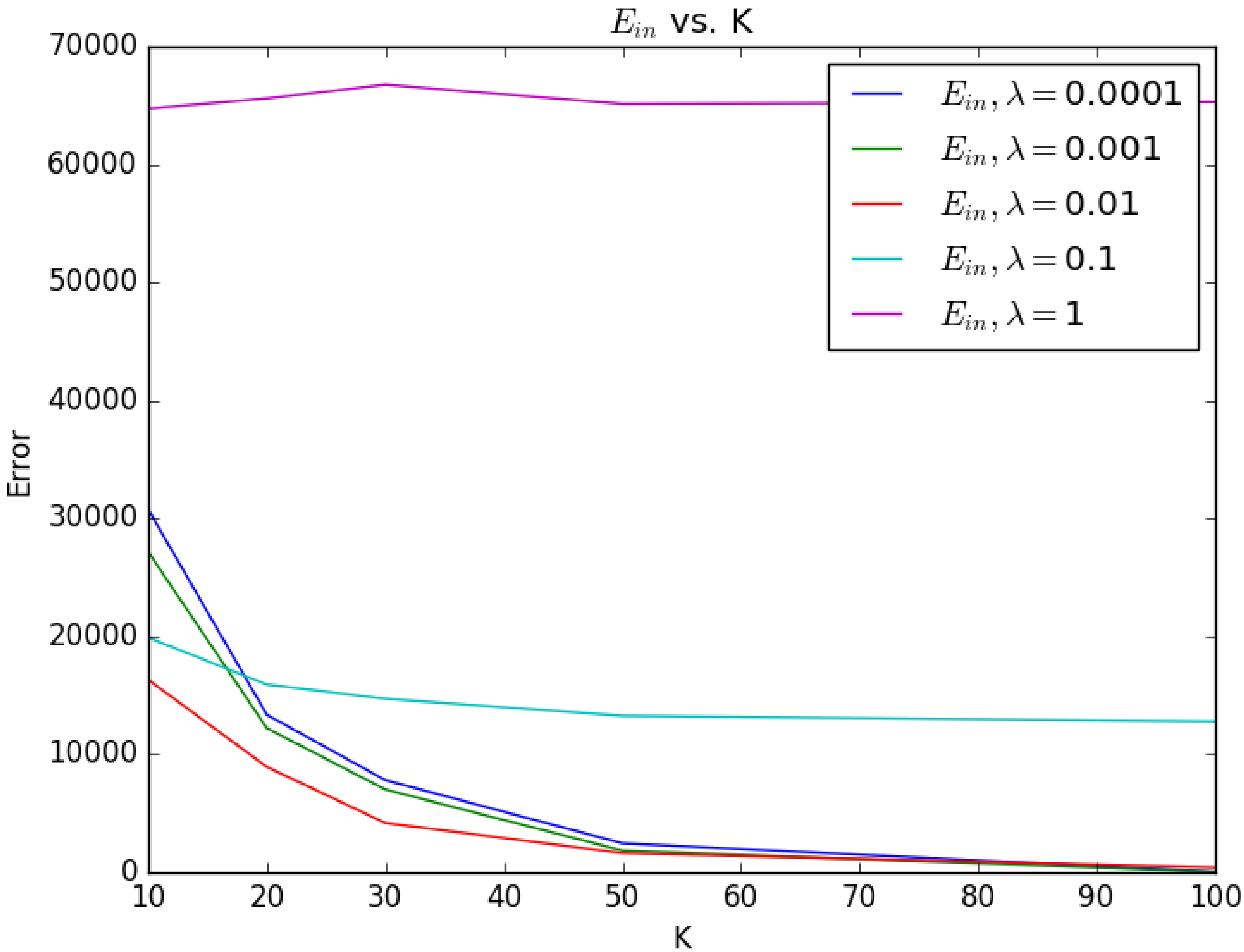
$\lambda v_j - \sum_i u_i^T y_{ij} + \sum_i u_i u_i^T v_j = 0$
$v_j(\lambda + \sum_i u_i u_i^T) - \sum_i u_i^T y_{ij} = 0$
$v_j = (\lambda + \sum_i u_i u_i^T)^{-1}(\sum_i u_i^T y_{ij})$

2.3 D

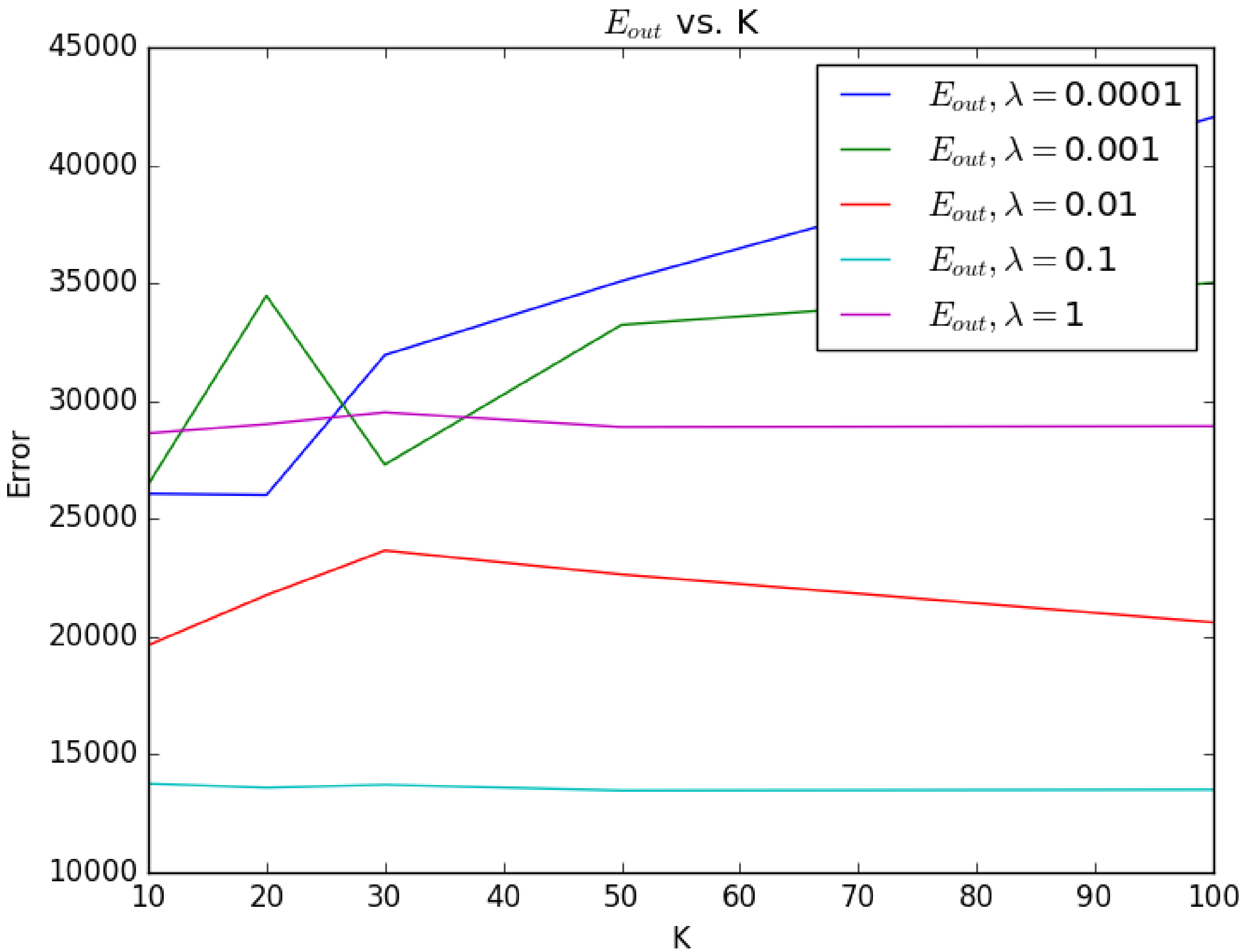


The out of sample error increased with K while the in sample error decreased. This is due to potential over-fitting that happens from the increased K values.

2.4 E



E_{in} vs. K
70000
60000
50000
40000
30000
20000
10000
0
Error
10
20
30
40
50
60
70
80
90
100
K
$E_{in}, \lambda = 0.0001$
$E_{in}, \lambda = 0.001$
$E_{in}, \lambda = 0.01$
$E_{in}, \lambda = 0.1$
$E_{in}, \lambda = 1$



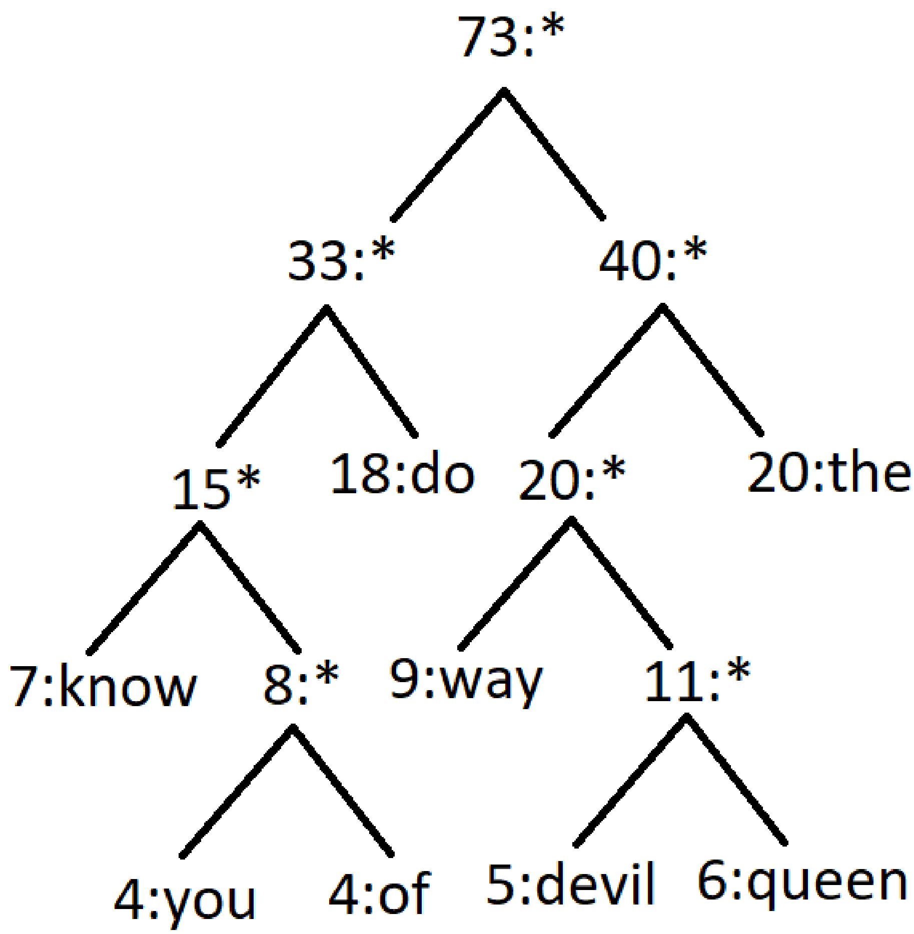
The lower lambda values generally lead to greater out of sample error values. The in sample errors give smaller with greater K values and are smaller for smaller lambda values. These trends are caused by overfitting, and regularization decreases overfitting.

3 Word2Vec Principles

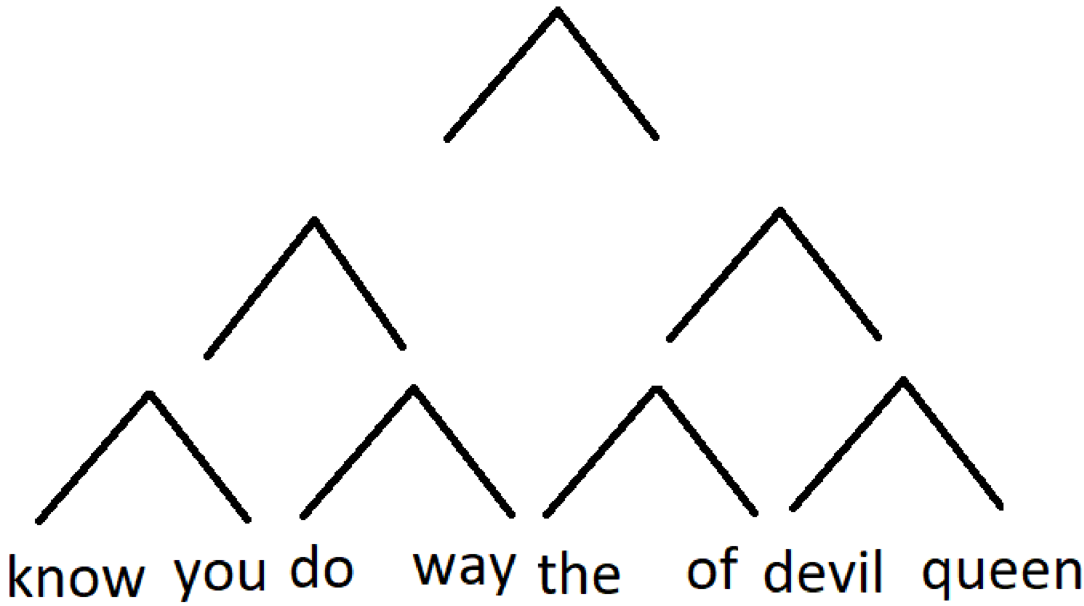
3.1 A

O(WD) The time complexity of the gradient calculations scales directly with both W and D.

3.2 B



Huffman tree (above)



Binary tree (depth 3, above)

The expected representation length averaged over the actual frequencies of the words in the Huffman tree is $\sum_{words}(pathlength)(frequency)$ which gives a value of 200/73, which is 2.740.

The expected representation length of the balanced binary tree is 3.

3.3 C

Larger D values would decrease the value of the training objective. The computation cost would also increase with a large D value, though.

3.4 E

3.4.1 i

Hidden layer weight dimension: 308 X 10

3.4.2 ii

Output layer weight dimension: 10 X 1

3.4.3 iii

Code output:
Pair(drink, thing), Similarity: 0.987077
Pair(thing, drink), Similarity: 0.987077
Pair(likes, wink), Similarity: 0.985708
Pair(wink, likes), Similarity: 0.985708
Pair(four, six), Similarity: 0.985455
Pair(six, four), Similarity: 0.985455
Pair(fish, black), Similarity: 0.984148
Pair(black, fish), Similarity: 0.984148
Pair(shoe, cold), Similarity: 0.982562
Pair(cold, shoe), Similarity: 0.982562
Pair(off, foot), Similarity: 0.982096

Pair(foot, off), Similarity: 0.982096
Pair(finger, top), Similarity: 0.980259
Pair(top, finger), Similarity: 0.980259
Pair(did, ever), Similarity: 0.979832
Pair(ever, did), Similarity: 0.979832
Pair(thin, slow), Similarity: 0.977941
Pair(slow, thin), Similarity: 0.977941
Pair(there, here), Similarity: 0.975115
Pair(here, there), Similarity: 0.975115
Pair(eight, nine), Similarity: 0.971293
Pair(nine, eight), Similarity: 0.971293
Pair(pink, wink), Similarity: 0.970619
Pair(teeth, gold), Similarity: 0.970029
Pair(gold, teeth), Similarity: 0.970029
Pair(mouse, fox), Similarity: 0.966468
Pair(fox, mouse), Similarity: 0.966468
Pair(green, ham), Similarity: 0.964749
Pair(ham, green), Similarity: 0.964749
Pair(glad, bad), Similarity: 0.963676

I noticed that a lot of the highest scores feature very "loyal" words; they only appear near each other and nowhere else. The pairs also tend to correspond to one another.