Homework 4 – Word Embedding

Discussion Session – 7PM Wednesday

Chetan Gandotra | cgandotr@ucsd.edu

Description of 100-dimension Embedding

For symmetric matrices, such as covariance matrices, we can find the eigenvalues/eigen vectors directly and use them to find the reduced dimension embedding using PCA. However, given that the matrix containing positive pointwise mutual information is asymmetric, I used Singular Value Decomposition (SVD) to obtain the 100-dimension embedding.

The initial (complete) positive pointwise mutual information matrix has dimensions 5000 x 1000, where each row represents a word in the 5000 vocabulary words set, and each column represents a context word *c*. Let us call this matrix X. Using SVD of Numpy, we get matrices U, S (by converting S given by Numpy to a diagonal matrix) and V, such that:

$$X = USV^T$$

Here, the dimensions of each of the matrix are as follows:

X → 5000 x 1000
U → 5000 x 5000
S → 5000 x 5000
V → 5000 x 1000

Note that since the column values were very similar in ranges, we did not have to perform normalization or preprocessing of data before running SVD. Also, note that S returned by Numpy is sorted in decreasing order.

To extract the 100-dimension embedding, we extract the first 100 columns from U, and the first 100 x 100 matrix (upper left part) from S, and take their multiplication. This matrix multiplication product gives us our new positive point wise mutual information matrix in reduced form, which will have dimensions of 5000 x 100. This matrix contains the first 100 principal components for all the 5000 vocabulary words. Hence, we used SVD to fetch the top *k* (100 in this case) features, without losing out on a lot of information. This procedure holds because the singular values are equal to the square root of eigen values, and the same principle as in PCA can be used for dimensionality reduction.

Thus, the above algorithm for finding the *k* dimensional embedding can be summarized as:

getKDimensionalEmbedding(X, *k*)

1. U, S, V ← numpy.linalg.svd(X)
2. S ← numpy.diag(S)
3. U ← U[:, :*k*] // Get first *k* columns
4. S ← S[:, :*k*] // Get first *k* columns
5. S ← S[:*k*] // Get first *k* rows
6. Return numpy.matmul(U, S) // U is n x k, S is k x k

Steps for finding V and C

To find the top 5000 words for vocabulary, and the top 1000 context words, the Brown corpus was downloaded and extracted into a local variable after removing stop words and punctuations contained in nltk.corpus.stopwords and string.punctuation respectively. Then, Python's Collections.Counter was used to extract the 1000 and 5000 most common (highest frequency) words for context words and vocabulary respectively.

The following pseudo-code demonstrates this procedure:

getVocabAndContextWords(vocab_size, context_size)

1. all_words ← brown.corpus()
2. words ← [ ]
3. cachedStopWords ← stopwords.words("english")
4. for *i* in range(len(all_words))
    a. currWord ← all_words[*i*].lower()
    b. if (not (currWord in cachedStopWords) and not (currWord in string.punctuation))
        i. for punc in string.punctuation
            1. currWord = currWord.replace(punc, '')
        ii. if not currWord
            1. words.append(currWord)
5. cnt ← Counter(words)
6. vocab ← [ ]
7. context_words ← [ ]
8. for *k*, *v* in cnt.most_common(vocab_size):
    a. vocab.append(*k*)
9. for *k*, *v* in cnt.most_common(context_size):
    a. context_words.append(*k*)
10. return vocab, context_words

Note that digits/numbers were not removed, because they are a prominent cluster of their own. This was indicative of the fact that the algorithm is doing well, as it places almost all number in the same cluster.

Nearest Neighbor Results

Twenty-five words, in addition to the fourteen words given in homework question were chosen, and their nearest neighbors were found using the cosine distance. The results are as follows:

Results on 14 given words:

On Reduced Matrix			On Original Matrix		
Word	**Closest Neighbor**	**Meaningful**	**Word**	**Closest Neighbor**	**Meaningful**
Communism	utopian	Yes	communism	utopian	Yes
Autumn	summer	Yes	autumn	summer	Yes
Cigarette	swung	No	cigarette	sleeping	No
Pulmonary	artery	Yes	pulmonary	artery	Yes
Mankind	world	Yes	mankind	language	Yes
Africa	asia	Yes	africa	asia	Yes
Chicago	portland	Yes	chicago	portland	Yes
Revolution	modern	Yes	revolution	movement	Yes
September	july	Yes	september	june	Yes
Chemical	drugs	Yes	chemical	drugs	Yes
Detergent	fabrics	Yes	detergent	fabrics	Yes
Dictionary	text	Yes	dictionary	text	Yes
Storm	night	Yes	storm	drove	No
Worship	religion	Yes	worship	degree	No

Table 1: Results for 14 Given Words with Original and Reduced Matrices

Results on 25 chosen words:

On Reduced Matrix [5000 x 100]			On Original Matrix [5000 x 1000]		
Word	**Closest Neighbor**	**Meaningful**	**Word**	**Closest Neighbor**	**Meaningful**
Artistic	imagination	Yes	artistic	literary	Yes
Dogs	horses	Yes	dogs	men	Yes
Temple	pistol	No	temple	distance	No
Lieutenant	killed	Yes	lieutenant	lawyer	Yes
michelangelo	moments	No	michelangelo	one	No
Democrats	elections	Yes	democrats	liberal	Yes
Advertising	sales	Yes	advertising	division	Yes
Missile	planning	Yes	missile	planning	Yes
rehabilitation	vocational	Yes	rehabilitation	provisions	No
Electronics	medical	No	electronics	procurement	Yes
Catholic	religious	Yes	catholic	church	Yes
Science	history	Yes	science	social	Yes
Literature	history	Yes	literature	history	Yes
Knowledge	thus	No	knowledge	development	Yes
Civilization	history	Yes	civilization	history	Yes
Shoulders	knees	Yes	shoulders	knees	Yes
Santa	easy	No	santa	occasionally	No
administrative	joint	Yes	administrative	hearing	Yes
Painting	works	No	painting	word	No
Chlorine	input	No	chlorine	cells	Yes
Weapon	machines	Yes	weapon	shot	Yes
Vision	person	Yes	vision	seems	No
submarines	systems	Yes	submarines	weapons	Yes
Institution	education	Yes	institution	college	Yes
Atlantic	union	No	atlantic	european	Yes

Table 2: Results for 25 Chosen Words with Original and Reduced Matrices

Analysis of results:

From the tables above, it is evident that the nearest neighbor technique using cosine distance does well in finding out the nearest neighbor of the given word. This is applicable to both the reduced matrix (5000 x 100) and the original matrix (5000 x 1000). There are a very few number of pairs that do not actually make sense in the reduced matrix. For example, in the above table, we have "atlantic – union" in the reduced matrix nearest neighbor results, which seems meaningless. On the other hand, the full matrix gives "atlantic – european", which seems like a better nearest neighbor, given that both the words in this pair can refer to regions. Also, for some values, reduced matrix nearest neighbor gives better results than the full matrix. This is evident in case of pairs like (lieutenant, killed) and (lieutenant, lawyer).

Clustering Results

Both the original positive pointwise mutual information matrix and the reduced matrix are not sparse but dense, due to which I used "K-Means with Elkan" algorithm, which uses Euclidean distance. K-means, being a "hard-clustering" algorithm, would have given more concrete results with the words not overlapping between various clusters, which is one of the main reasons I chose to use it. The Elkan algorithm uses triangular inequality to speed up convergence and was hence preferred. Note that whether the matrix is sparse or not was found using

Python's scipy.sparse.issparse API. The cluster centers were initialized using k-means++ method, which avoids poor clusters.

The algorithm was run for 100,000 iterations, and gave very good results. The words in most of the clusters seemed very coherent and similar. Here are a few examples (limited the number of words in each cluster):

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
months	walter	18	excitement	tax	service	machine
hours	lawrence	24	gesture	amount	military	aircraft
summer	jackson	16	delight	return	labor	engine
month	thompson	13	warmth	bill	services	flight
winter	clark	100	frightened	due	staff	vacation
session	harry	inches	trembling	income	economy	weapon
days	johnson	diameter	mysterious	gross	personnel	camera
week	samuel	pounds	disturbed	payment	management	mechanical

Cluster 8	Cluster 9	Cluster 10
world	countries	wife
life	communist	mother
god	nations	son
death	union	father
spirit	soviet	husband
st	leaders	marriage
christ	germany	friend
lord	berlin	baby
jesus	cuba	brother

Table 3: Clustering Results

Properties of clusters:

- Cluster 1 → Relates to time units/seasons
- Cluster 2 → Relates to names (proper nouns)
- Cluster 3 → Relates to numbers and units of measurement
- Cluster 4 → Relates to feelings and personality traits
- Cluster 5 → Relates to income, revenue and taxes
- Cluster 6 → Relates to military and management
- Cluster 7 → Relates to mechanical stuff and machines
- Cluster 8 → Relates to religion
- Cluster 9 → Relates to countries
- Cluster 10 → Relates to human relations

Note that there are many other clusters which give good results and I have picked only the first few to write down here. Looking at the results we can safely infer that the clustering algorithm gives good results.