Spectral Graph-Based Methods for Learning Word Embeddings

Paul Trichelair
Master 2 MVA
Ecole Normale Supérieure de Paris-Saclay
ptrichel@esn-paris-saclay.fr

Souhail Toumdi
Master 2 MVA
Ecole Normale Supérieure de Paris-Saclay
stoumdi@ens-paris-saclay.fr

Abstract

Word embeddings have become a widely used technique in Natural Language Processing to perform machine learning tasks on textual data. In the past few years, many previous works on these representations have reported good experimental results. To date, most algorithms use some variants of neural networks[1] or standard spectral methods like Principal Component Analysis (PCA)[2] or Canonical Correlation Analysis (CCA)[3]. In this paper, we focus on more recent graph-based approaches to learn word embeddings from word co-occurrence graphs. We provide a review of three well known methods along with their implementations and results on some artificial data.

1 Introduction

It is obvious that every mathematical system or algorithm needs some sort of numeric input to work with. For this reason, many applications of machine learning, such as part-of-speech tagging and joint representations for images and text, require numerical representations of textual data known as word embeddings. These representations can carry useful information about the word actual meaning. In fact, semantically similar words are mapped to nearby points with respect to a measure of similarity. To build such representations, one could look at the context of a word in a sentence (neighbouring words). Today, most algorithms are based on this assumption and use different techniques (Neural network, Principal Component Analysis, Canonical Correlation Analysis,...) and they proved to be successful for such tasks.

In this paper, we discuss three spectral graph-base methods for learning word embeddings: complete isometric feature mapping, linear locally embedding and laplacian eigenmaps. Those methods are well known for non-linear dimensionality reduction. In fact, based on co-occurrence matrix of words, the idea is to learn low-dimensional embeddings that preserves local distances and geometry of the manifold formed by these words in the high-dimensional space. We start by introducing the motivations and the algorithms of the three methods before using them to learn word embeddings of some artificial data.

2 Graph-based methods for learning word embeddings

2.1 Complete isometric feature mapping: Isomap

Isomap is one of several widely used low-dimensional embedding methods. It extends Multidimensional scaling method[1] by preserving the intrinsic geometry of the data. Multidimensional scaling (MDS) performs low-dimensional embedding based on the pairwise distance between data points and it results in a spatial arrangement that preserves distances. It is equivalent to PCA when the used metric is the euclidean distance. However, Isomap is based on the same steps but differs in

the construction of the matrix of distances between data points. It is based on the geodesic distance that captures the underlying manifold more accurately than using the euclidean distance allowing to detect non-linear structures that are invisible for classical MDS. In fact, for neighbouring data points, euclidean distance provides a good approximation of the geodesic distance but for faraway points, It can be approximated by summing up a sequence of short walks according to the edges of a predefined graph. Figure 1 illustrates how Isomap exploits the geometry of the data to construct low-dimensional embeddings. In fact, two neighbour points in the original data are also neighbours in the recovered embedding.

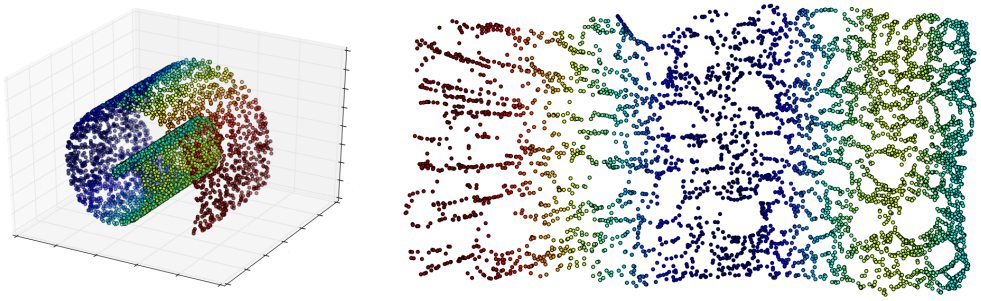


Figure 1: Swiss roll data (a) and it's two dimensional embedding plot (b) generated using Isomap (our implementation)

The Isomap algorithm can be divided into 3 major steps:

1. Build a weighted connected graph over the data points where the weight w_{ij} corresponds to the distance between points i and j in the input space X, $d_X(i,j)$. We don't keep all the edges of the graph: two simple methods are to connect each point only to all points within some fixed radius ϵ, or to all of its K nearest neighbors.
2. Estimate the geodesic distances between points in the input space using shortest-path distances on the constructed graph. The shortest-path distance problem is the problem of finding a path between two vertices in a graph which minimizes the sum of the weights of its constituent edges. For weighted undirected graphs, the most popular algorithm for finding shortest-paths is **Dijkstra's** [2] algorithm that we will use in the implementation later. Another efficient algorithm was designed by *Floyd* [3]. First, we initialize the coefficients of the resulting matrix $W_G(i,j)$ with $d_X(i,j)$ if i and j are connected in the graph and with ∞ otherwise. Then, for each value $k \in \{1,...,N\}$, we replace all entries $W_G(i,j)$ by $\min\{W_G(i,j), W_G(i,k) + W_G(k,j)\}$. The resulting matrix W_G will contain the shortest-path distances between all pairs of points.
3. The last step consists on applying a classical MDS [1] on the resulting matrix W_G constructing an embedding of the data in a d-dimensional euclidean space Y that best preserves the manifold's estimated intrinsic geometry. The intuition behind this algorithm is to find data points in a d-dimensional space whose Euclidean distances equal these geodesic distances. The algorithm that performs this task is the following:

Algorithm 1 Multidimensional scaling

Require: $D \in \mathbb{R}^{n\times n}$ $(D_{ii} = 0, D_{ij} \geq 0)$, $d \in \{1,...,n\}$
- Set $B = -\frac{1}{2}HDH$ where $H = I_n - \frac{1}{n}\mathbb{1}\mathbb{1}^T$ is the centering matrix.
- Compute the spectral decomposition of B: $B = U\Delta U^T$.
- Set $X = U\Delta^{\frac{1}{2}}$.
- Return $[X]_{n\times d}$.

We remark from the description of the algorithm that the only parameter is the one used in the graph's construction (k or ϵ). The dimensionality of the underlying manifold is automatically estimated. In fact, *Theorem 1* in [4] proves that the eigenvalues found by MDS are non-negative and the dimensionality is given by the number of non-zero eigenvalues. However, there will be some noise

and uncertainty in the estimation process, so one looks for a tell-tale gap in the spectrum to determine the dimensionality.

2.2 Locally Linear Embedding: LLE

Locally Linear Embedding [5] is based on a totally different intuition that doesn't involve computing pairwise geodesic distances between widely separated data points. It is based on a simple observation. Consider a manifold embedded in a high dimensional space. If the manifold is sufficiently smooth, one could approximate the later as a collection of overlapping small patches that are linear. Hence, each data point and its neighbors will lie on locally linear patch of the manifold that is characterized with some weights allowing to reconstruct each data point from a convex combination of its neighbors. The weights are chosen to minimize the following reconstruction error for each point i

$$||x_i - \sum_{j \in \mathcal{N}(i)} W_{ij} x_j||_2$$

From the equation above, we can extract two major characterization of the matrix W. In fact, each row must sum to one (convex combination) and $W_{ij} = 0$ if $j \notin N(i)$ which reflects the local character of this method. We note that W is invariant to global rotations and scalings thanks to the constraints and the optimization problem. Moreover, it locality preserves the geometry of the manifold and distances between pairs of points allowing to include non-linear characteristics when computing the embeddings in a lower dimensional space. Once W is computed, the embedding configuration could be found by solving the following minimization problem with respect to y_i

$$\sum_i ||y_i - \sum_j W_{ij} y_j||_2$$

The cost function can be rewritten as $Y^T M Y$ where $M = (I - W)^T (I - W)$. There are a couple of constraint on Y. First, we force $Y^T Y = I_d$ so the solution is of rank d. Second, $\sum_i Y_{ij} = 0$ forces the resulting embeddings to be centred on the origin.

Using Lagrange multipliers, we come to a closed form of the solutions of the two optimization problems as described in the following LLE algorithm.

Algorithm 2 Locally Linear Embedding

Require: $x_1, x_2, ..., x_n \in \mathbb{R}^p$, $d \in \{1, ..., p\}$ the dimensionality of the embedding and k the number of neighbours to use.

- Compute reconstruction weights. For each point x_i, set

$$W_{ij} = \frac{\sum_k C_{kj}^{-1}}{\sum_{l,m} C_{lm}^{-1}}$$

where $C_{kj} = (x_i - \nu_j)^T (x_i - \nu_k)$, ν_j and ν_k are two neighbors of x_i

- Compute the low-dimensional embeddings. Let U be the matrix whose columns are the eigenvectors of $(I - W)^T (I - W)$ with nonzero accompanying eigenvalues.
- Return $[U]_{n \times d}$.

We can notice that the main parameter in the algorithm is k, the number of neighbours that we use to reconstruct each point. Figure 2 shows the result obtained for the Swiss roll data using 12 neighbours.

2.3 Laplacian Eigenmaps

The Laplacian Eigenmaps is based on ideas from spectral graph theory. In fact, given a graph and a matrix of weights W, the Laplacian of the graph L reveals a wealth of information about the graph. This method use the Laplacian to capture local information about the manifold. The main idea of the Laplacian Eigenmaps is to express the preservation of the local information as an optimization

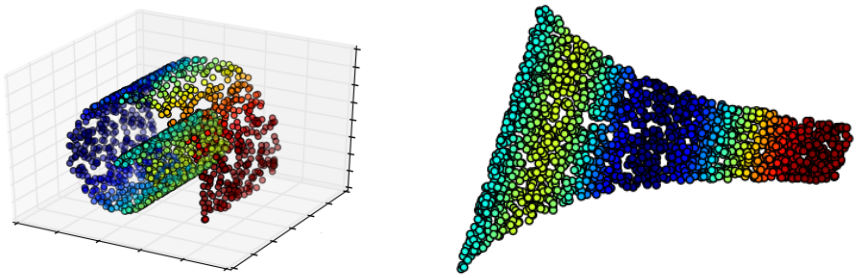


Figure 2: Swiss roll data (a) and it's two dimensional embedding plot (b) generated using LLE (our implementation)

problem that involves the weight matrix and the embedding map we are looking for. We want the points that are close in the high dimensional space to have similar embedding values. The notion of similarity between two points in the high dimensional space is related to the weight matrix W. Therefore, given the embedding $Y = [Y_1, Y_2, .., Y_m]$ the problem we try to solve can be formulated as:

$$\min_Y \sum_{i,j} ||Y_i - Y_j||^2 W_{ij} \iff \min_Y tr(Y^T L Y)$$

with L the Laplacian of the graph. This problem is minimized when all $Y_i = 0$. To avoid this configuration, a constraint of $Y^T D Y = I$ is added that fix the scale at which Y_i's are placed. One can show that the solution to this problem is to find the eigenvectors of the Laplacian Random Walk associated with the m smallest eigenvalues.

The Laplacian Eigenmaps algorithm can be divided into 3 major steps:

1. Build the adjacency graph like for Isomap. Two simple methods are to connect each point only to all points within some fixed radius ϵ, or to all of its K nearest neighbors.
2. Compute the weights for connected components using *Heat function*. If two nodes i and j are connected then compute

   $$W_{ij} = \exp\left(-\frac{||x_i - x_j||_2^2}{t}\right)$$

   with t a parameter to choose.
3. Compute the eigenvalues and eigenvectors of the Laplacian Random Walk.

   $$L_{rw} = I - D^{-1}W$$

   where W the weight matrix and D the degree matrix.
4. Return the eigenvectors corresponding to the $m + 1$ smallest eigenvalues. Leave out the eigenvector corresponding to the eigenvalue 0 that indicates that the graph is connected.

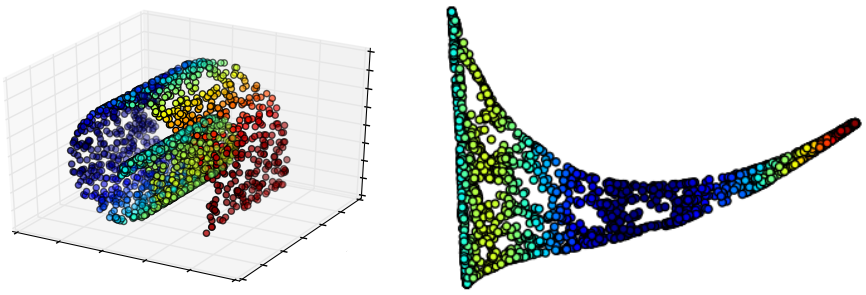


Figure 3: Swiss roll data (a) and it's two dimensional embedding plot (b) generated using Laplacian Eignemaps (our implementation)

3 Experimental part

In this section, we provide a qualitative evaluation of the embeddings generated using the exposed graph-based algorithms. We use co-occurrence matrix of 10 000 words as a input data to our algorithms. We study two different size of context: 2 and 5. First, we study two methods of normalization of the input data, TF-IDF and PPMI, and explain its importance. We will also explain the method of evaluation before comparing the three methods against several state-of-the-art embeddings: GLOVE [6], Word2Vec [7] and HPCA [8].

3.1 Co-occurrence matrix' normalization

Simple frequency in co-occurrence matrices isn't the best measure of association between words and it isn't discriminative. Stop words for example are not informative about any particular word as they can be found in the context of every word. So, we would like context words that are particularly informative about the target word. This can be done through the normalization of the co-occurrence matrix. We start by introducing a couple of methods of normalization.

- **Positive pointwise Mutual Information PPMI** : it is a well known measure in information theory and it quantifies how often two pointwise mutual information events x and y occur, compared with what we would expect if they were independent. In the context of a co-occurrence matrix $F = (f_{ij})$ where f_{ij} represents the co-occurrence of the word i in the context j, the PPMI is computed using the following formula:

$$\text{PPMI(word,context)} = \max\left(0, \log\left(\frac{\mathbb{P}(\text{word,context})}{\mathbb{P}(\text{word})\mathbb{P}(\text{context})}\right)\right)$$

  Where the probabilities are

$$\mathbb{P}(\text{word,context}) = \frac{f_{\text{word,context}}}{\sum_i \sum_j f_{ij}} \qquad \mathbb{P}(\text{word}) = \frac{\sum_j f_{\text{word},j}}{\sum_i \sum_j f_{ij}} \qquad \mathbb{P}(\text{word}) = \frac{\sum_i f_{i,\text{context}}}{\sum_i \sum_j f_{ij}}$$

  The numerator tells us how often we observed **word** in **context** while the denominator tells us how often we would expect **word** in **context** assuming they are independent.

- **Term frequency-Inverse document frequency tf-idf** [9]: it's another common normalization's method. In the context of a collection of documents, a first step consists simply on computing the frequency of each word. The second step consists on penalizing words that appear in most of the documents because they are not helpful for characterization. Consequently, terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection. In the context of the same matrix F, we consider each context as a separate document and we apply tf-idf's method. The first step is already computed as f_{ij} represents the frequency of the word i in the context j. The second step is performed using the following formula:

$$\text{tf-idf(word,context)} = f_{\text{word,context}} * \log\left(\frac{\text{\# of contexts}}{\text{\# of contexts where } \textbf{word} \text{ appears}}\right)$$

3.2 Evaluation method

To evaluate the embeddings we use several methods:

1. **Analogy :** It's relative to natural semantic and syntactic relations in the embedding. For instance we want to have a proximity between a capital and its corresponding country but also between an adjective and its corresponding superlative. To test it, we use a dataset that contains a list of pair of words relative to each syntactic and semantic relations. The relation is verified if we find the two words in the same neighborhood. To check it we used the 10 nearest neighbors of one of the two words.
2. **Relatedness :** We use three datasets, WordSim-353, Rg65 and Rareword, that contains human relatedness scores for pair of words. We first compute, based on our embeddings, the cosine similarity for each pair of word contained in the dataset. To try to see if the embedding captures the same relation between two words, we compute the correlation of the cosine similarity with human relatedness score. An high correlation is related to an embedding that behave the same as human to depict similarity along pair of words.
   Two types of correlation are computed :

- The Pearson product-moment correlation coefficient that measures linear correlation between two variables. It's the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$$

- The Spearman correlation. It's defined as the Pearson correlation coefficient between the ranked variables.

$$r_s = \frac{\text{cov}(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}}$$

  Using this method the correlation can be non affine.

All the models we used embedded words into 50 and 200 dimensional spaces ($D = 50, D = 200$). For a fixed dimension, we vary the number of neighbours K in the range $\{10, 20, 50, 100, 200, 500\}$. For Laplacian Eigenmaps, we vary the factor t of the heat function in the range $\{5, 25, 10000\}$. We therefore computed 24 embeddings for LLE and isomap and 72 for Laplacian eigenmaps for each normalization. We report only the best performance obtained for each evaluation criterion. All the results could be found in the joined Excel sheet.

3.3 Evaluation of the graph based embeddings

The results obtained using embeddings' dimension $D = 200$ are always better than the ones obtained with $D = 50$. In fact, as we increase the dimension, we capture more information about the structure of the manifold and non-linearity in the data. First, we would like to compare the impact of the size of the context on the results. Appendix 1 show plots of the different criteria computed for each embeddings method and and for each size of the context. For GLOVE, Word2vec and HPCA, the size of the context doesn't have a huge impact on the results. However, for Isomap and Laplacian Eigenmaps, a context size of 2 seems to give better results in average while a context size of 5 in more suitable for LLE method.

Concerning the normalizations, we get better results for Laplacian Eigenmaps and LLE when using PPMI normalization while TF-IDF normalization gives better results for Isomap.

In the following we will only report the best results obtained for each criterion when using the three graph-based algorithms and we compare them to the three state-of-the-art methods. These results are summarized in table 1.

Criterion	GLOVE	Word2Vec	HPCA	Isomap	LLE	Lap.Eig
Semantic Analogy micro	85.89%	**91.65%**	75.83%	**68.08%**	66.51%	51.66%
Semantic Analogy macro	**76.34%**	76.09%	68.57%	**65.62%**	64.37%	46.12%
Syntactic Analogy micro	**88.32%**	83.08%	74.52%	54.78%	**83.08%**	52.39%
Syntactic Analogy macro	**82.33%**	78.9%	68.29%	42.99%	**74.90%**	39.79%
Rareword pearson	**0.571**	0.546	0.463	0.360	**0.466**	0.390
Rareword spearman	**0.592**	0.564	0.486	0.427	**0.569**	0.451
Rg65 pearson	**0.826**	0.764	0.657	0.703	**0.725**	0.704
Rg65 spearman	**0.816**	0.765	0.740	0.712	**0.754**	0.701
wordsim353 pearson	**0.643**	0.602	0.520	0.597	**0.572**	0.516
wordsim353 spearman	**0.661**	0.628	0.522	0.614	**0.617**	0.554

Table 1: Comparison of the graph based model with state of the art methods

As shown in Table 1, comparing only the three graphs-based methods, LLE gives the best results in average, followed by Isomap. In fact, LLE performs significantly better than Isomap and Laplacian Eigenmaps on the syntactic task. Its performance on this task in comparable to GLOVE an Word2vec and better than HPCA. For the others tasks, LLE and Isomap perform similarly and worse than the others methods. If we increase the dimension of the embeddings, Isomap reaches a performance that is comparable to the three state-of-the-art methods on the Semantic task (Table2). However, increasing the dimension could lead to computational problems when using a corpus of high cardinality.

Criterion	Isomap (d = 1000)
Semantic Analogy micro	80.12%
Semantic Analogy macro	70.01%
Syntactic Analogy micro	55.80%
Syntactic Analogy macro	46.52%
Rareword pearson	0.389
Rareword spearman	0.521
Rg65 pearson	0.654
Rg65 spearman	0.606
wordsim353 pearson	0.546
wordsim353 spearman	0.595

Table 2: Isomap results for embeddings' dimension of 1000

3.4 Coherence task

The evaluation of a word embedding is not easy. Our evaluation was only based on a set of predefined scored for word pairs. With the coherence task, we try to see if groups of words in a small neighborhood in the embedding space are related.
To do so, people usually use Amazon mechanical turk, displaying a small set of words extracted from the same neighborhood and introducing one intruder. If the embedding is good, the intruder is easy to detect.
In this report, we compute the neighborhood around one frequent word "cost" and one rare word "householder" and try to see if we can make a choice based on this set of words. We also try to correlate those results to the figures obtained in the previous part.

Method	Analogy mean	Neighborhood
L. E. k=20 t=25	20%	cost, demand, profits, expenses, guaranteed, paying
Isomap k=200	30%	cost, costs, prices, wages, expenses, inflation, substantial
L. E. k=500 t=25	45%	cost, demand, costs, spending, price, deficit, proportion, capacity
Isomap k=500	47%	cost, costs, demand, expense, prices, substantially
LLE k=50	56%	cost, costs, expense, burden, cash, savings, revenue, expenses, worth, budget
LLE k=200	73%	cost, costs, expense, burden, cash, revenue, savings, worth, expenses, budget

Table 3: Neighborhood of "cost" in the embedding obtained for different methods with dimensionality 200 and a context size of 2

First, we can observe that the set of words is always meaningful regardless of the method. Even the embedding related to an a score of 20% gives interpretable results. It's impossible for someone to choose between the best embedding based on the set of neighbors.
Let's try to the same experiment with a less frequent word: "householder".

Method	Analogy mean	Neighborhood
L. E. k=20 t=25	25%	householder, residing, socorro, xs, beginning, kitt
Isomap k=200	30%	householder, xs, socorro, residing, beginning, in
L. E. k=500 t=25	45%	householder, socorro, kitt, xs, beginning, households, spacewatch
Isomap k=500	47%	householder, xs, socorro, residing, beginning, in, since, fontsize
LLE k=50	56%	householder, couples, martial, closely, impression, eurovision, register, iucn, households, dnp, pga
LLE k=200	73%	households, median, couples, householder, makeup, socorro, kitt, rv, wake

Table 4: Neighborhood of "householder" in the embedding obtained for different methods with dimensionality 200 and a context size of 2

We can notice that most of the words are meaningless. None of the embeddings succeed to perform well.
For the word "householder" a mix a several embedding could provide a list that is a bit more meaningful. We can think of it as a good strategy to obtain better results.

Choosing between those embeddings based on the previous tables is not at all obvious. We don't see any correlation with the scores obtained before. A good strategy to choose a word embedding could be to repeat this procedure with a larger list of words.
We represented below the neighborhood of "cost" for k=50.

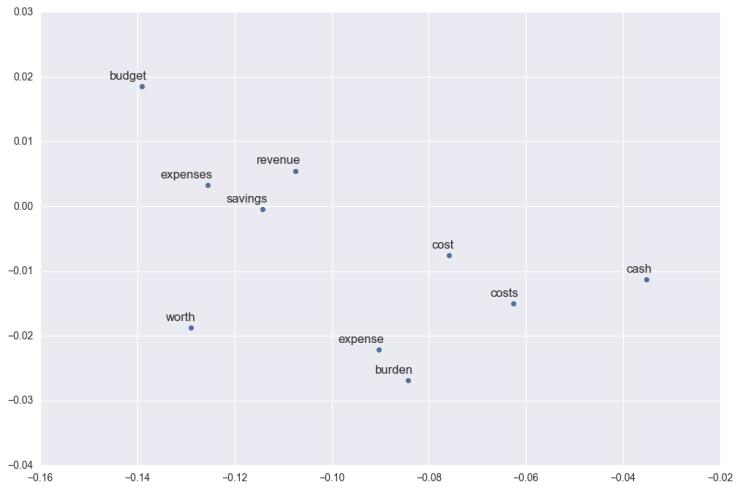


Figure 4: Neighborhood obtained for k=50 in the embedding space obtained with locally linear embedding

4 Conclusion and further thinking

In this report, we study 3 non-linear dimensionality reduction methods to build words embeddings based on the matrix of co-occurrence of words. These methods are based on modelling the interactions between samples within a fully connected graph. The evaluation have shown that the best embeddings' method is LLE followed by Isomap. Each of them is efficient on some tasks. However, the results are still relatively poor comparing them to the other well known methods. One way to enhance them is to combine the different embeddings.
The evaluation of an unsupervised word embedding is not easy. The little test realized in this report shows that we can extract relevant information from embedding related to low scores. Therefore, we can think that the real issue is to find a relevant method to evaluate unsupervised word embeddings.

References

[1] T.F. Cox and M.A.A. Cox. Multidimensional Scaling. Chapman and Hall/CRC, 2nd edition, 2001.

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik. 1: 269–271. 1959.

[3] Robert W. Floyd. Algorithm 97: Shortest Path. Communications of the ACM. 5 (6): 345. 1962.

[4] L. Cayton. Algorithms for manifold learning. 2005

[5] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. 2000.

[6] J. Pennington, R. Socher, C. D. Manning. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.

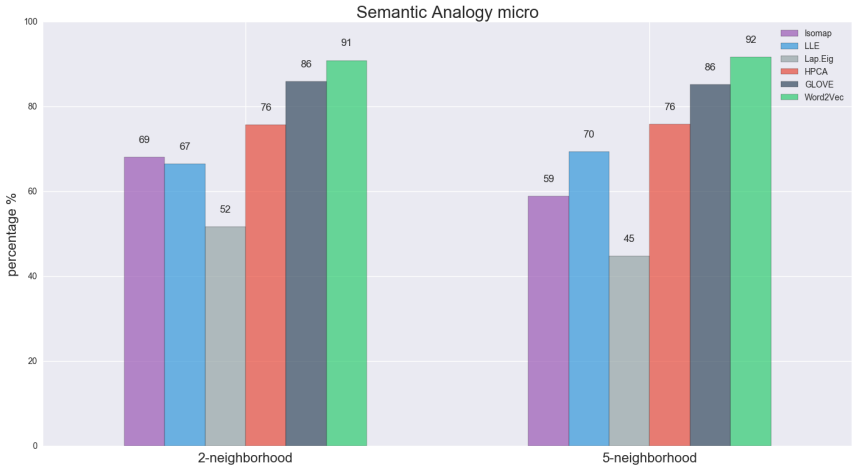
[7] T. Mikolov and al. Distributed Representations of Words and Phrases and their Compositionality. Advances in neural information processing systems, 3111-3119.

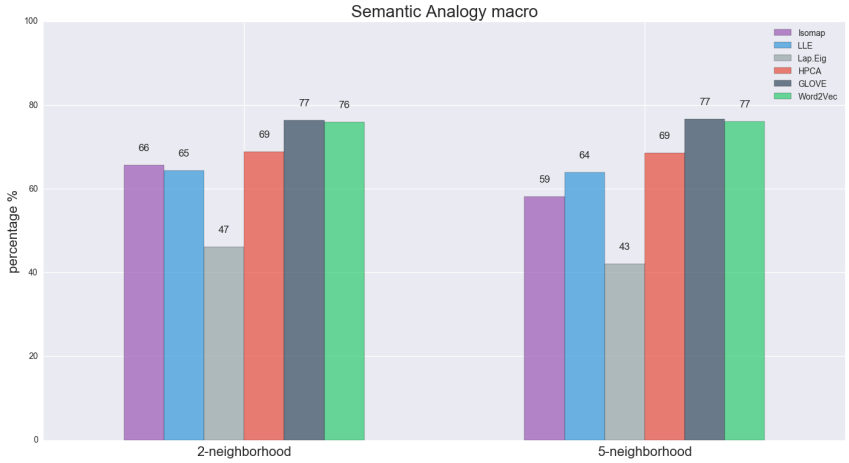
[8] R. Lebret and R. Collobert. Word Embeddings through Hellinger PCA. 2014.

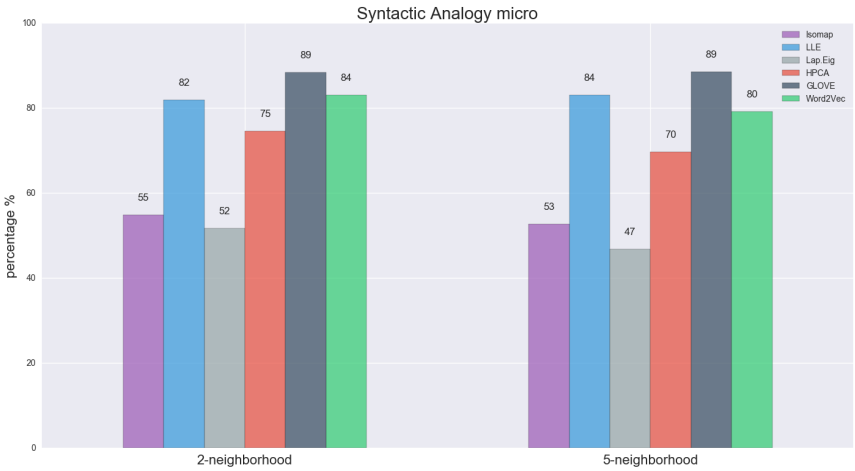
[9] J.Ramos.Using TF-IDF to Determine Word Relevance in Document Queries.

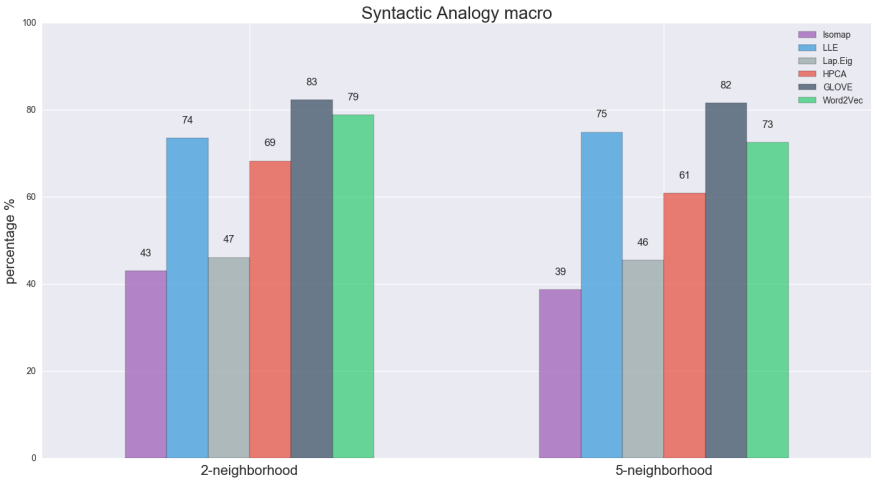
Appendices

Appendix 1

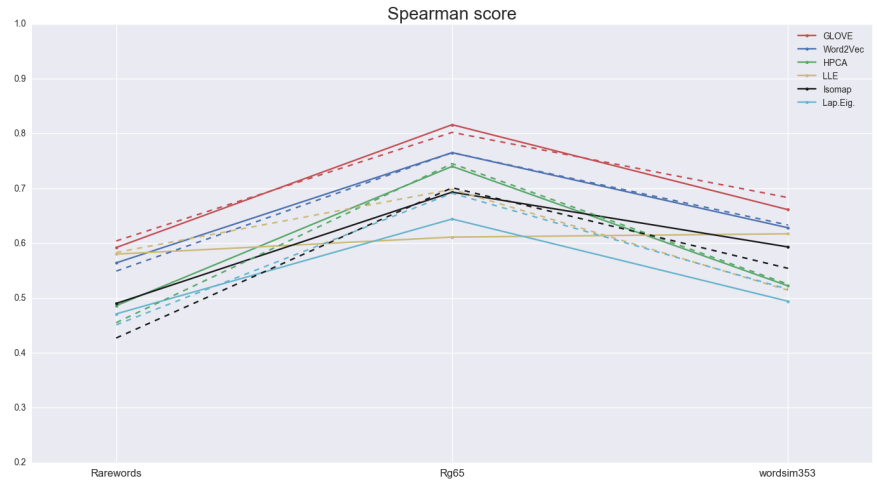







Syntactic Analogy macro
percentage %
100
80
60
40
20
0
Isomap
LLE
Lap.Eig
HPCA
GLOVE
Word2Vec
43
74
47
69
83
79
39
75
46
61
82
73
2-neighborhood
5-neighborhood


Pearson score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
GLOVE
Word2Vec
HPCA
LLE
Isomap
Lap.Eig.
Rarewords
Rg65
wordsim353


Spearman score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
GLOVE
Word2Vec
HPCA
LLE
Isomap
Lap.Eig.
Rarewords
Rg65
wordsim353