Information Retrieval

Latent Semantic Indexing

Hamid Beigy

Sharif university of technology

December 15, 2018

Table of contents

Term-document matrix

1. Consider the following term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95

2. This matrix is the basis for computing the similarity between documents and queries.
3. Can we transform this matrix, so that we get a better measure of similarity between documents and queries?

Table of contents

Latent semantic indexing (an overview)

1. We will decompose the term-document matrix into a product of matrices.
2. We use singular value decomposition (SVD).
3. SVD is $C = U\Sigma V^T$ (where C= term-document matrix)
4. We will then use the SVD to compute a new, improved term-document matrix C'.
5. We'll get better similarity values out of C' (compared to C).
6. Using SVD for this purpose is called latent semantic indexing or LSI.

Example of $C = U\Sigma V^T$

1. Consider the following term-document matrix

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

2. This is a standard term-document matrix.
3. Actually, we use a non-weighted matrix here to simplify the example.

Example of $C = U\Sigma V^T$

1. The matrix U equals to

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

2. One row per term, one column per $\min(M, N)$ where M is the number of terms and N is the number of documents.
3. This is an orthonormal matrix: (i) Row vectors have unit length. (ii) Any two distinct row vectors are orthogonal to each other.
4. Think of the dimensions as semantic dimensions that capture distinct topics like politics, sports, economics.
5. Each number u_{ij} in the matrix indicates how strongly related term i is to the topic represented by semantic dimension j.

Example of $C = U\Sigma V^T$

1. The matrix Σ equals to

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

2. This is a square, diagonal matrix of dimensionality $min(M, N) \times min(M, N)$.
3. The diagonal consists of the singular values of C.
4. The magnitude of the singular value measures the importance of the corresponding semantic dimension.
5. We'll make use of this by omitting unimportant dimensions.

Example of $C = U\Sigma V^T$

1. The matrix V^T equals to

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

2. One column per document, one row per $\min(M, N)$ where M is the number of terms and N is the number of documents.
3. This is an orthonormal matrix: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.
4. These are again the semantic dimensions from matrices U and Σ that capture distinct topics like politics, sports, economics.
5. Each number v_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension j.

Latent semantic indexing (Summary)

1. We've decomposed the term-document matrix C into a product of three matrices: $U\Sigma V^T$.
2. The term matrix U – consists of one (row) vector for each term
3. The document matrix V^T – consists of one (column) vector for each document
4. The singular value matrix Σ – diagonal matrix with singular values, reflecting importance of each dimension
5. Next: Why are we doing this?

Table of contents

How we use the SVD in LSI

1. Key property: Each singular value tells us how important its dimension is.
2. By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
3. These details may
   - be noise – in that case, reduced LSI is a better representation because it is less noisy.
   - make things dissimilar that should be similar – again, the reduced LSI representation is a better representation because it represents similarity better.
4. Analogy for "fewer details is better"
   - Image of a blue flower
   - Image of a yellow flower
   - Omitting color makes is easier to see the similarity

Reducing the dimensionality to 2

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Actually, we only zero out singular values in Σ.
This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$.

Reducing the dimensionality to 2

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Original matrix C vs. reduced $C_2 = U\Sigma_2 V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

We can view C_2 as a two-dimensional representation of the matrix C. We have performed a dimensionality reduction to two dimensions.

Why the reduced matrix C_2 is better than C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

- Similarity of d_2 and d_3 in the original space: 0.
- Similarity of d_2 and d_3 in the reduced space: ≈ 0.52
- boat and ship are semantically similar.
- The reduced similarity measure reflects this.
- What property of the SVD reduction is responsible for improved similarity? (Do it as an exercise.)

Table of contents

Why we use LSI in information retrieval?

1. LSI takes documents that are semantically similar (=talk about the same topics),
2. But these documents are not similar in the vector space (because they use different words)
3. LSI re-represents them in a reduced vector space in which they have higher similarity.
4. Thus, LSI addresses the problems of synonymy and semantic relatedness.
5. Standard vector space: Synonyms contribute nothing to document similarity.
6. Desired effect of LSI: Synonyms contribute strongly to document similarity.

How LSI addresses synonymy and semantic relatedness?

1. The dimensionality reduction forces us to omit a lot of detail.
2. We have to map different words to the same dimension in the reduced space.
3. The cost of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
4. SVD selects the least costly mapping (see below).
5. Thus, it will map synonyms to the same dimension.
6. But it will avoid doing that for unrelated words.

Comparison to other approaches

1. Recap: Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.
   or, more commonly, too few terms in common for a high similarity score
2. LSI increases recall and hurts precision. (why? do as an exercise.)
3. Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion and it has the same problems.

Implementation

1. Compute SVD of term-document matrix
2. Reduce the space and compute reduced document representations
3. Map the query into the reduced space $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$.
4. This follows from: $C_k = U_k \Sigma_k V_k^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$
5. Compute similarity of q_k with all reduced documents in V_k.
6. Output ranked list of documents as usual
7. Exercise: What is the fundamental problem with this approach?

Optimality

1. SVD is optimal in the following sense.
2. Keeping the k largest singular values and setting all others to zero gives you the optimal approximation of the original matrix C (Eckart-Young theorem)
3. Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates C better.
4. Measure of approximation is Frobenius norm: $||C||_F = \sqrt{\sum_i \sum_j c_{ij}^2}$
5. So LSI uses the best possible matrix.
6. There is only one best possible matrix – unique solution
7. There is only a tenuous relationship between the Frobenius norm and cosine similarity between documents. (describe it as an exercise.)

Table of contents

Language modeling

1. An language model is a model for how humans generate language.
2. The quality of language models is measured based on their ability to learn a probability distribution over words in vocabulary V.
3. Language models generally try to compute the probability of a word w_t given its $n-1$ previous words, i.e. $p(w_t|w_{t-1}, \cdots w_{t-n+1})$.
4. Applying the chain rule and Markov assumption, we can approximate the probability of a whole sentence or document by the product of the probabilities of each word given its n previous words:

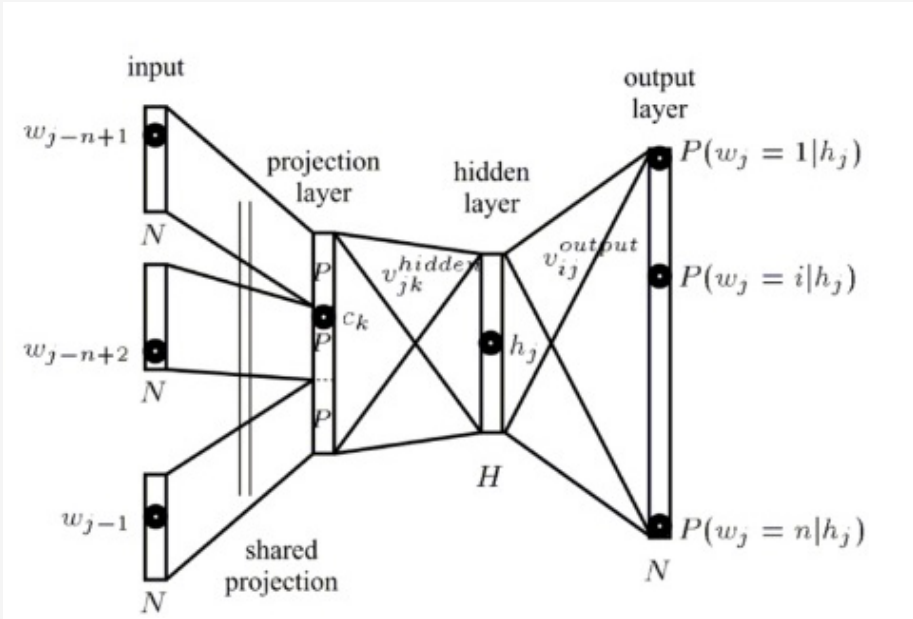
$$p(w_1, \cdots, w_T) = \prod_i p(w_i|w_{i-1}, \cdots, w_{i-n+1})$$

5. In n-gram based language models, we can calculate a word's probability based on the frequencies of its constituent n-grams:

$$p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{count(w_{t-n+1}, \cdots, w_{t-1}, w_t)}{count(w_{t-n+1}, \cdots, w_{t-1})}$$

Neural Probabilistic Language Model

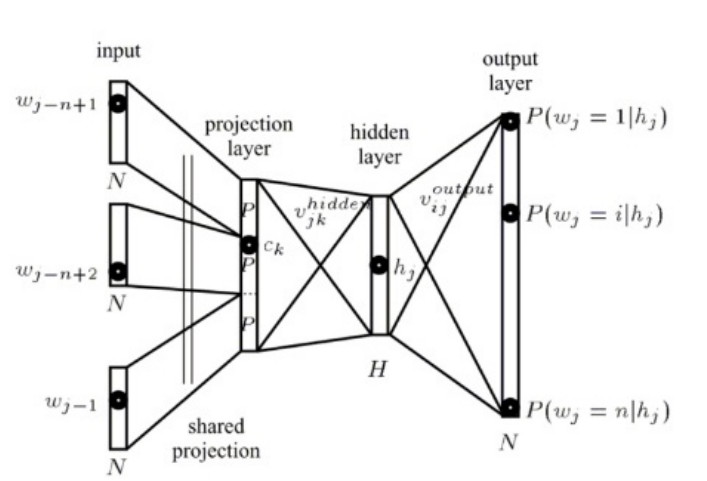
1. In NNs, we achieve the same objective using the softmax layer



$$p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

Neural Probabilistic Language Model

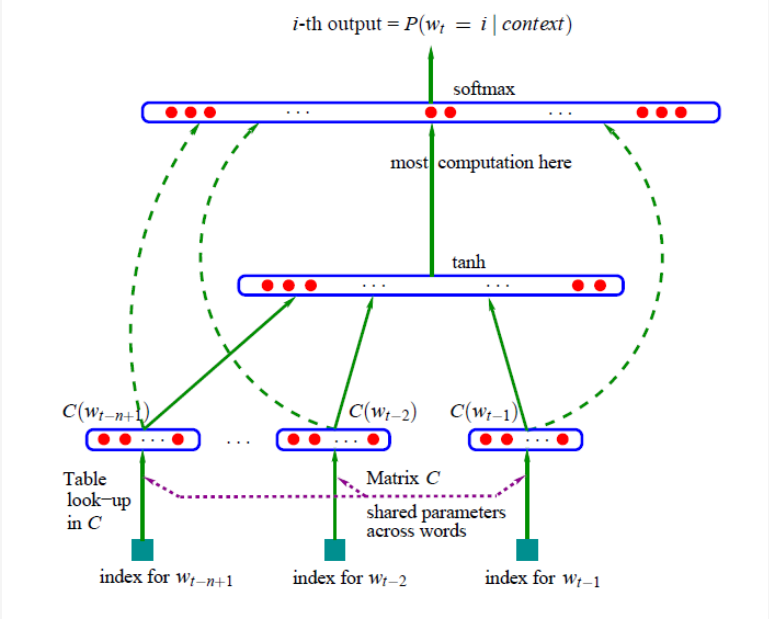
1. In this model



2. The inner product $h^\top v'_{w_t}$ computes the (unnormalized) log-probability of word w_t , which we normalize by the sum of the log-probabilities of all words in V.
3. h is the output vector of the penultimate network layer, while v'_w is the output embedding of word w, i.e. its representation in the weight matrix of the softmax layer.

Neural Probabilistic Language Model

1. In NNs, we achieve the same objective using the softmax layer



2. Associate each word in vocabulary a distributed feature vector.
3. Learn both the embedding and parameters for probability function jointly.

Table of contents

Word2vec algorithm

1. Proposed by Mikolov et. al. and widely used for many NLP applications (in two papers).
2. Key features
   - Uses neural networks to train word / context classifiers (feedforward neural net)
   - Uses local context windows (environment around any word in a corpus) as inputs to the NN
   - Removed hidden layer.
   - Use of additional context for training LMs.
   - Introduced newer training strategies using huge database of words efficiently.

Word2vec algorithm

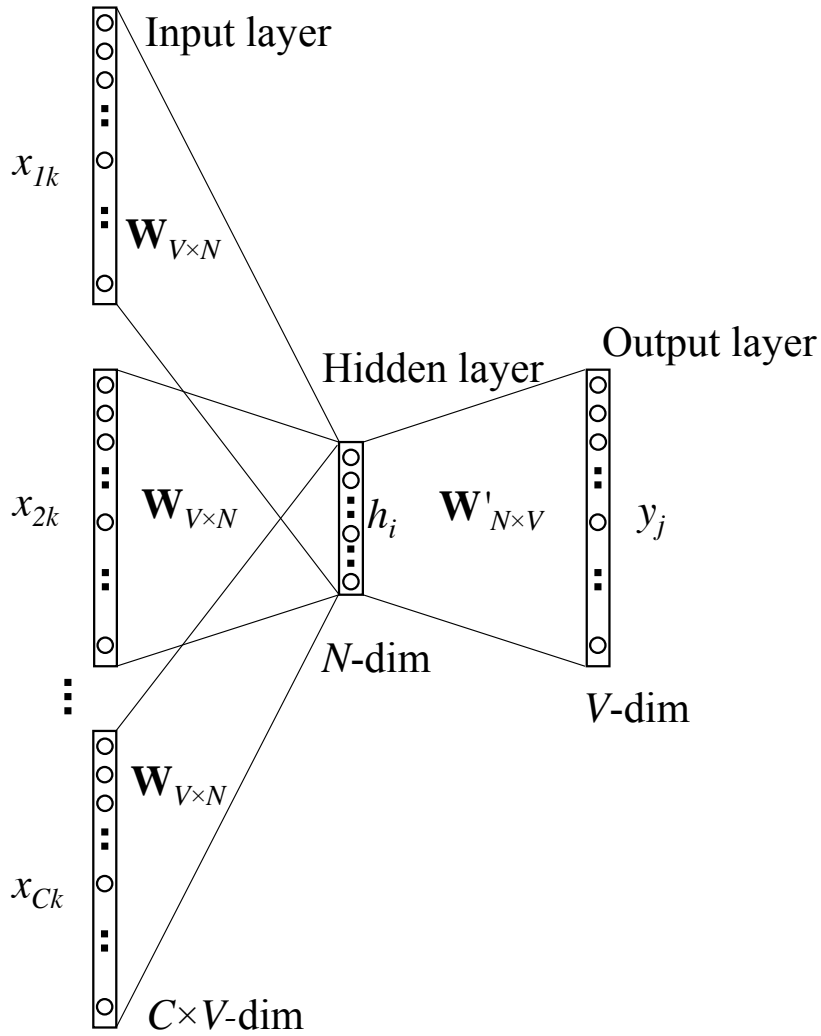
1. In their first paper, Mikolov et al. propose two architectures for learning word embeddings that are computationally less expensive than previous models[1].
2. In their second paper, they improve upon these models by employing additional strategies to enhance training speed and accuracy[2].

[1] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1-12.

[2] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1-9.

Continuous Bag-of-Words

1. Mikolov et al. thus use both the n words before and after the target word w_t to predict it.
2. They call this continuous bag-of-words (CBOW), as it uses continuous representations whose order is of no importance.



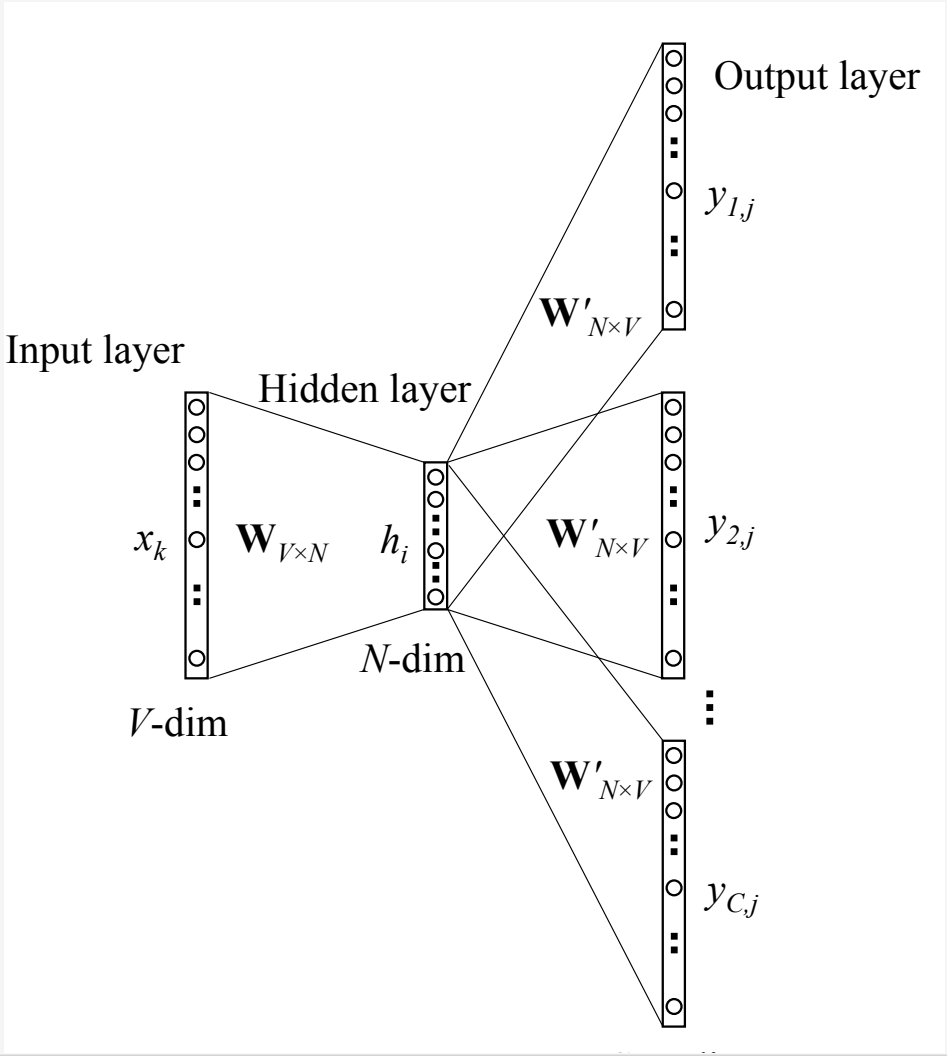
Continuous Bag-of-Words objective function

1. The objective function of CBOW in turn is

$$l(\theta) = \sum_{t \in Text} \log P(w_t | w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

Skip-gram

1. Instead of using the surrounding words to predict the center word as with CBOW, skip-gram uses the centre word to predict the surrounding words.



Skip-gram objective function

1. The skip-gram objective thus sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t to produce the following objective function.

$$I(\theta) = \sum_{t \in Text} \sum_{-n \leq j \leq n, \neq 0} \log P(w_{t+j} | w_t)$$

How represent documents?

1. We represent a document using
   - Average of its vectors
   - Weighted average of its vectors
   - Sentence2 Vec
   - Paragraph2Vec
   - Doc2Vec
   - ⋮

Table of contents

Reading

Please read section 18 of Information Retrieval Book.