Word2Vec using Negative Constrastive Estimation

Julina Maharjan, jmaharja@kent.edu

Kent State University – KSU

November 25, 2019

Content

Content

Motivation

Key Idea in NLP

- is how can we efficiently convert words into numeric vectors
- such that it can then be fed into various machine learning models to perform predictions

Technique

- **Word2Vec**

Motivation

Why do we need Word2Vec?

Convert the words into some set of numeric vectors

A straight-forward way

To use a "one-hot" vector
i.e converting the word into a sparse representation with only one element of the vector set to 1, the rest being zero.

Example: For the sentence " the cat sat on the mat" would have the following vector representation.

$$\begin{bmatrix} the \\ cat \\ sat \\ on \\ the \\ mat \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Motivation

Dimension of the new matrix is **6 x 5**
and, the size of the vocabulary = 5

what if words are huge?

the input layer into NN will have at least 10,000 nodes
such that it will strips away any local context of the words - information about closely appearing words will be loss

Therefore, an efficient way that conserves information is word2Vec

Word2Vec- Information preserving

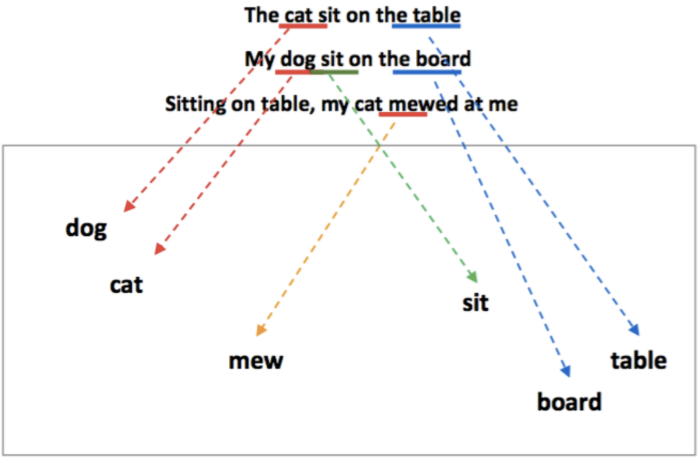


Figure: Similar words clustering in the same space

Neural Network Perspective

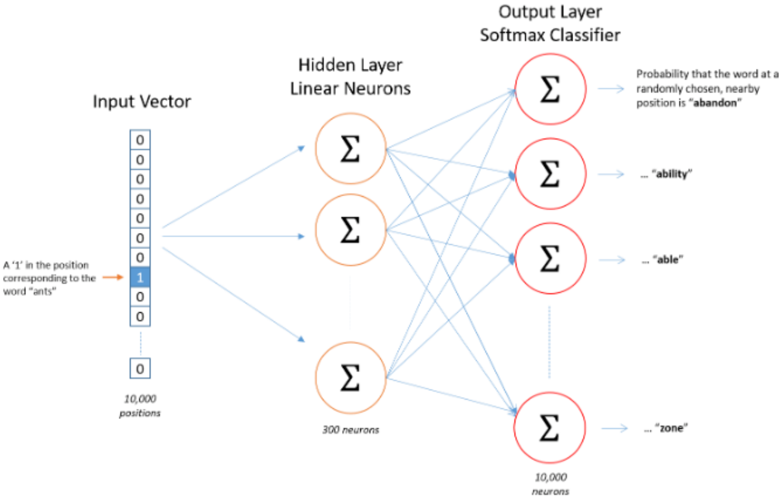


Figure: The architecture of word2vec Neural network

2 Components

I. Word Embedding

The first is the mapping of a high dimensional one-hot style representation of words to a lower dimensional vector.
for instance, transforming a 10,000 columned matrix into a 200 columned matrix

II. Finding the probability of each word

The second is to maintain the word context i.e meaning
Two way of doing this:

1. CBOW approach
2. Skip-gram approach (more famous because it produces more accurate results on large datasets)

Embedding and Probability layer

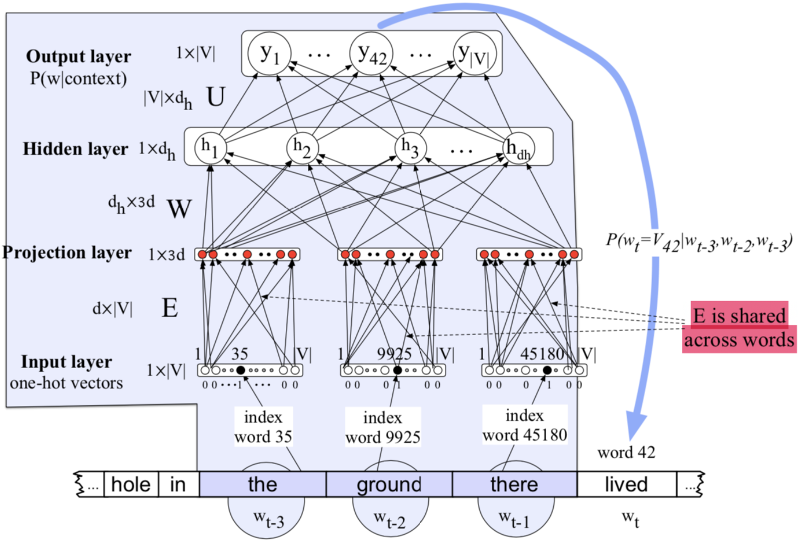
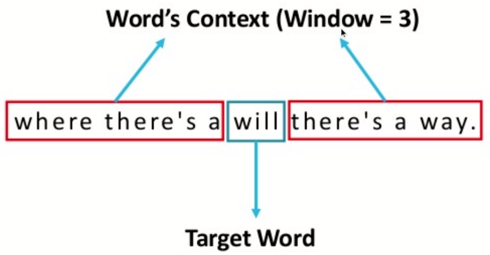


Figure: Similar words clustering in the same space

Layers in Mathematical notation

$$
\begin{aligned}
e &= (Ex_1, Ex_2, \ldots, Ex) \\
h &= \sigma(We + b) \\
z &= Uh \\
y &= \mathrm{softmax}(z)
\end{aligned}
$$

Word's context



Skip-Gram Model

Skip-gram

This model predicts the probabilities of a word being a context word for the given target

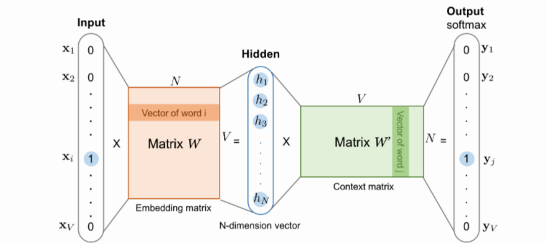


Figure: The Skip-Gram Model

Skip-gram Model Example

Example

"The man who passes the sentence should swing the sword. Ned Stark

Sliding window (size =5)	Target Word	Context
[The man who]	the	man,who
[The man who passes]	man	the,who,passes
[The man who passes the]	who	the,man,passes,the
[man who passes the sentence]	passes	who, the, sentence
...	...	...
[should swing the sword]	the	should,swing,sword
[swing the sword]	sword	swing,the

Continuous Bag-of-Words (CBOW

predicts the target word (i.e "swing") from source context words.

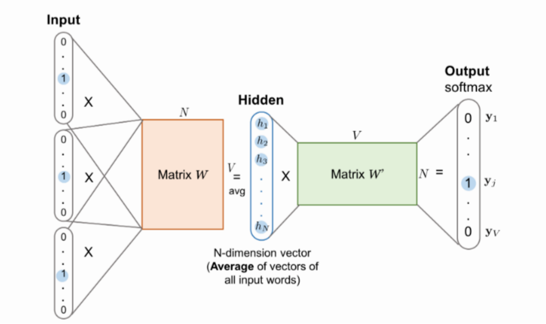


Figure: The CBOW Model

Content

Limitations

Softmax Function

In skip gram model, the probability of a word **w** given context **c** is

$$p(w_O|w_I) = \frac{exp(v'_{w_O} T_{v_{w_I}})}{\sum_{w=1}^{W} exp(v'_{w_O} T_{v_{w_I}})}$$

Short Comings in Softmax function

Have to compute probabilistic expression over the corpus – a computationally intensive task

Solutions

1. noise contrastive estimation
2. negative sampling

both avoid the full summation over the corpus

NCE

- Instead of calculating a probability distribution over all possible target words, NCE uses logistic regression to distinguish a target from samples from a noise distribution.

Negative Sampling

- Negative sampling (used in word2vec code), also learns the parameters of the model as a binary classification problem (every time a word is tugged closer to its neighbors, it is also tugged away from k samples picked from a unigram distribution).

Content

NCE

- a new **estimation principle** for parameterized statistical models
- the idea is to perform **logistic regression** to discriminate between the observed data and some artificially generated noise
- works well for **unnormalized models**, i.e models where the density function does not integrate to one.
- simulations show that NCE offers the **best trade-off** between computational and statistical efficiency

Review of Logistic Regression

- Logistic regression can be used to obtain a classifier which discriminates between the data sets $\mathbf{X} = x(1), ..., x(T))$ and $\mathbf{Y} = y(1), ..., y(T))$
- Logistic regression uses the model

$$P(\mathbf{u} \in \mathbf{X}; \theta) = \frac{1}{1 + \exp(-G(\mathbf{u}; \theta))}$$
$$P(\mathbf{u} \in \mathbf{Y}; \theta) = 1 - P(\mathbf{u} \in \mathbf{X}; \theta)$$

  where $G(\mathbf{u}; \theta)$ is a function parameterized by θ

- For

$$G(\mathbf{u}; \theta) > 0, P(\mathbf{u} \in \mathbf{X}; \theta) > 0.5$$

  and the input u is classified to belong to $\mathbf{X}$.
- For a linear classifier: $G(\mathbf{u}; \theta) = w_0 + \mathbf{w}^T \mathbf{u}$
  Parameters θ are $\{w_0, \mathbf{w}\}$

Learning By Comparison

- Assume we know the properties of Y (noise).
- We let classifier to learn the difference between X and Y.
- From the learned difference between X and Y, we can thus deduce properties of X.
- This can be formalized using estimation theory

NCE

- Observe data $\mathbf{X} = (\mathbf{x}(1), \ldots, \mathbf{x}(T))$ with unknown pdf p_d
- Generated noise $\mathbf{X} = (\mathbf{y}(1), \ldots, \mathbf{y}(T))$ with known pdf p_n
- Define a parameterized function $f(\mathbf{u}; \theta)$, which models the data log-density $\log p_d(\mathbf{u})$
- Use logistic regression with the non linearity

$$G(\mathbf{u}; \theta) = f(\mathbf{u}; \theta) - \log p_n(\mathbf{u})$$

- Conditional likelihood leads to the objective function

$$J(\theta) = \sum_t \log[h(\mathbf{x}(t); \theta)] + \log[1 - h(\mathbf{y}(t); \theta)]$$

$$\text{where } h(\mathbf{u}; \theta) = \frac{1}{1 + \exp[-G(\mathbf{u}; \theta)]}$$

- The estimator is defined as $\hat{\theta} = \operatorname{argmax} J(\theta)$

Properties of Estimators

- Assume the parametric model $f(u;\theta)$ can approximate any function. Then, the maximum of objective J is attained when

$$f(\mathbf{u};\theta) = \log p_d(\mathbf{u})$$

  where $p_d(u)$ is the pdf of the observed data
- **Corollary:**
  For data generated according to model, i.e.

$$\log p_d(\mathbf{u}) = \log p_m(\mathbf{u};\theta^*)$$

  we can show that the estimator is statistically consistent.
- Supervised learning thus leads to unsupervised estimation of a probabilistic model given by log-density $f(u;\theta)$

Content

Conclusion

- Instead of predicting the next word (the "standard" training technique), the optimized classifier simply predicts whether a pair of words is good or bad.
- **Consistent** nature of NCE gives the best approximation of softmax for word2Vec
- Simulations shows that NCE is very **fast** and converges to accurate solution by $1/\sqrt{n}$

References I

Hyvarinen Aapo.
Estimation of non-normalized statistical models by score matching.
Journal of Machine Learning Research, 6(35):695–709, 2005.

Gutmann Michael and Hyvarinen Aapo.
Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.
Journal of Machine Learning Research, 135(35):13096–13106, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
Journal of Machine Learning Research, 135(35):13096–13106, 2013.

Word2Vec using Negative Constrastive Estimation

Julina Maharjan, jmaharja@kent.edu

Kent State University – KSU

November 25, 2019