NLPIE

- Vectors based on term-frequency (counts of words in a corpus wrt to each document)
- Vectors based on Positive Pointwise Mutual Information (PPMI)

Chararteristics:
  - Sparse vectors (large no. of zero entries)
  - Large size vectors (length of vector is equal to vocabulary size |V|)
  - However, we can reduce the size using dimensionality reduction techniques such as PCA, SVD,etc

Word2Vec
- uses a technique called as skip gram
- negative sampling

Aim: Instead of counting or calculating some values the values, train a classifier.

Task: Prediction - Is a target word t is likely to show up near the context c.
However, we skip the prediction and te weights of the learned classifier will be used as embeddings.

For a classifier, we require a supervised or labelled training data.

The running text from the corpus can be coonsidered as supervised training data.

Skip gram follows the steps as:

1) Treat the target word t and its neighbouring context c as positive sample.
2) Randomly use the other words in the vocab to obtain the negative samples.
3) Train the classifier such as logistic regression.
4) Use the regression weights as embeddings.

The Classifier:

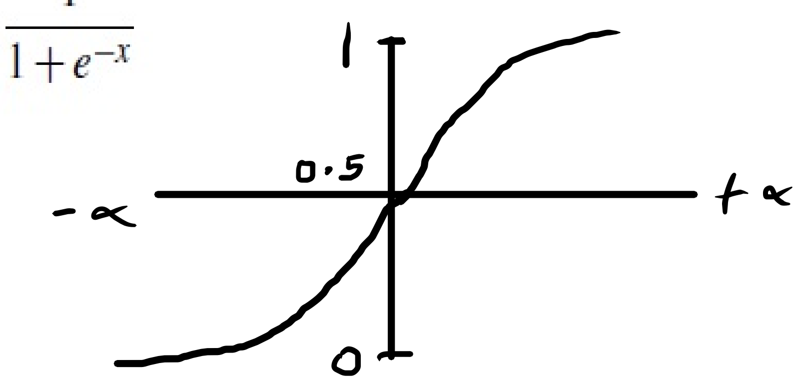
Train classifier such that the tuple (t,c), generates the probabilities as

P(+ | t,c)
P(- | t,c) = 1 - P(+ | t,c)

A target word is likely to appear near the context, if its embedding is similar to the context embedding.

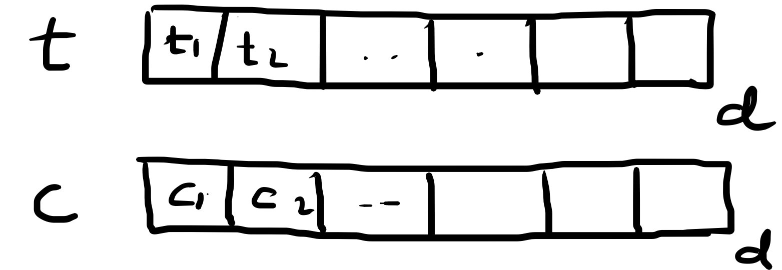
Similarity(t,c) == t . c (dot product) // between - infinity and + infinity

We bound this similarity using the logistic function $\sigma(x) = \dfrac{1}{1+e^{-x}}$



$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$t \cdot c = t_1 * c_1 + t_2 * c_2 + \dots t_d * c_d$



$$P(-|t,c) = 1 - P(+|t,c)$$
$$= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}}$$

when $t_1, t_2 \dots t_d$ and $c_1, c_2 \dots c_d$ are randomly generated values.

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

$$\log P(+|t,c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-t\cdot c_i}}$$

Since all the context words of target word are independent of each other we can multiply the probabilities.

$$L(\theta) = \sum_{(t,c)\in+} \log P(+|t,c) + \sum_{(t,c)\in-} \log P(-|t,c)$$

While training

Goal:

1) Maximize the loss of positve samples.
2) Minimize the loss of negative samples.

Optimizer like gradient descent may be used to optimize the weights or specifically, the randomly generated embeddings.