Feature Representation

Word Embeddings: Represening text/words as a real valued vectors.

Distributional Hypothesis: Words that occur in similar contexts have similar meanings.

Distributional Hypothesis is the idea behind word embeddings. When the vectors representations are mapped, words with similar meaning are grouped together.

Word Embedding Algorithms:

1. Embedding Layer
2. Word2Vec
   1. Continuous Bag-of-Words or CBOW model
   2. Continuous Skip-Gram model
3. GloVe - Global Vectors to Word Representation

Pre-trained word embeddings are available for free that can be used directly in the task at hand. However, the kind of usage varies:

* Static - use the word embeddings as is as a component of model.
* Updated - use the word embeddings as initial values(or word embeddings of an auxilary task) and re-train them using the model.

Coding:

Do: Webscrape and pre-process the data, make it ready for input to word2vec algorithm.

Continuous Skip-gram model or Skip-gram mode with negative sampling (SGNS)

A binary classifier is trained to predict or to answer a question - Is word w likely to show up near target word t. In anothers the classifier needs to predict of ‘w’ is part of ‘t’’s context. For training with negative samples, data is chosen such that ‘w’ and ‘t’ are not from the neighborhood - negative sampling. Below is the intuition of SGNS:

* Treat ‘t’ and ‘w’ from same neighborhood as positive samples.
* Treat ‘t’ and ‘w’ from different neighborhood as negative samples. Both positive and negative samples should come from same corpus though.
* Use logistic regression to differentiate beween 2 samples. Notice that NN is not needed here.
* Use the regression weights as embeddings. In another words use regression weight vectors as new representation of the words.

The Classifier

Goal is to train the classifier such that, give a tuple(‘t’, ‘c’), it will return the probability of how similar is the center word the target word. Coefficients or weights learned during the classification task are the actual word embeddings we are after. High probability for positive samples and low probability for negative samples.

Probability the ‘c’ is context word for ‘t’ is given by:

P(+|t,c)

And that they both are not from same neighborhood is:

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

Lets take a step back. What does the probability measure here?

Probability here gives a measure of target word ‘t’ and context word ‘c’ belonging to same context. As per distributional hypothesis, words occuring in the same or similar context are similar. So probability gives the measure of similarity between pair of target and context words.

So how similarity is measured?

Similarity between 2 vectors are similar if their dot product is high - cosine similarity.

How come in order to learn word embeddings, we need word embeddings?

We initialize the word embeddings with small values.

How high should the dot product be?

The dot product could lie anywhere between negative infinity and positive infinity. Hence the dot product is scaled to lie between [0, 1] using sigmoid or logstic function.

sigmoid(t.c) = 1/(1+exp(-t.c))

P(+|t,c) = 1 / (1+exp(-t.c))

P(-|t,c) = exp(-t.c) / (1+exp(-t.c))

Now those equations are for one target and context word pair. Since skip gram operates over a window and underlying skip gram assumption is that the words are independent from each other. So the above probability for k context words can be expressed as