Café, bonne nuit

Have a coffee when you run any code, if it is still running, then have two….

THEORY

TensorFlow 06: Word Embeddings (1)

Posted on June 14, 2016

(A new learner! Leave comments if you find any mistakes please!)

How do we present a word? In TensorFlow, everything, yes everything, flows into the graph, is a tensor. For example, the famous **word2vec** model is used for learning vector representations of words.

Representation

It is natural that we will use RGB channels to represent an image pixel by pixel. So we

I care about…

Algorithm (16)

Deep Learning (21)

Energy-Based Learning (2)

will represent an image in a huge tensor, contains those channels with proper values for colors. What about other signals, e.p. audios and human movements. They all can be seen as functions on time-series. We can represent them as a tensor with the cleaned true values directly. When it comes to words, we want to find a similar way to represent them as well.

One goal for Deep Learning is to learn representations (called representation learning). That is, let's find some ways to represent something. For example, suppose we are going to build an image search engine. If we find out a way to transfer every image into a value, e.p, image A is 3, image B is 5, image C is 300. Closing value means similar images. When we put image A as a query, we will find image B instead of image C, because B is similar. Now we are representing images with real values, so you see how powerful it is!

Sparse Text Representation

Bag-of-Words (BoW) model. Consider the task to classify documents. The core is to find out how similar two documents are, then you could use algorithms like K-means to find out. We would first know how many

Graph Database (11)

Inference Models (2)

Machine Learning (14)

Natural Language Processing (6)

Problem Shooting (1)

Python (16)

Statitics (9)

Theory (18)

Uncategorized (2)

Gallery

words in our corpus. Then we could represent a document based on how many time each word in corpus occurs as a vector. Finally, we would have an occurrence matrix. Another process contains stop-words removing, TF-IDF, etc. However, the matrix will be very sparse. If you have a universal corpus which has words from all domains, a domain-specific document would contain only a small part of the huge corpus. That means you will have lots of zeros in the vector of occurrence. The sparse problem could be a motivation for word representation.

Word Representation

Usually to represent a word, we will consider its context. There is a classic example of thinking of:
Sentence A “I use TensorFlow to study Deep Learning”.
Sentence B “Lee uses Caffe to study Deep Learning”.
We have reasons to believe that “TensorFlow” and “Caffe” have similar meaning because they appearance in the same position and structure.

CBOW Understanding (repost part from a Chinese blog)

Posts

June 2016

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

« May Jul »

Predict a word w, given its context Context(w). So with word w, it is a positive sample, while other words are all negative samples. When given a positive sample $(Context(w), w)$, the function defined as below:

$g(w) = \prod_{u \in \{w\} \quad \cup NEG(w)} p(u|Context(w))$

It is simply a multiplication of all conditional probabilities of a target word, given the current context. So we wish to maximize it. Where,

$$p(u|Context(w)) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 1; \\ 1 - \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 0, \end{cases}$$

We define a "flag" function L^w, when word u is in a positive sample, we let $L^w(u) = 1$; otherwise, it is 0. And the $\sigma(X_W^T \theta^w)$ gives the probability of predicting target word w, given Context(w). If we put everything into function g(w):

$g(w) = \sigma(X_W^T \theta^w) \prod_{u \in NEG(w)} 1 - \sigma(X_W^T \theta^u)$

So the first part denotes the probability of the positive sample, while the second one denotes the probabilities of negative ones. We would like to maximize function g(w), which means we will maximize the value of $\sigma(X_W^T \theta^w)$ (positive) and at the same time minimize $\sigma(X_W^T \theta^u)$ (negative). That would make the positive and negative samples to be distinguish. We could take log of it for

future computing.
Keep in mind that the parameter θ is to be optimized. Do not cover the details here.

Skip-gram Model Understanding

In skip-gram, it is a reversed version of CBoW. So the task is we will The skip-gram Model is mainly based on negative sampling. Skip-gram and CBoW are somehow similar

Similar objective function can be found in the TF tutorial:

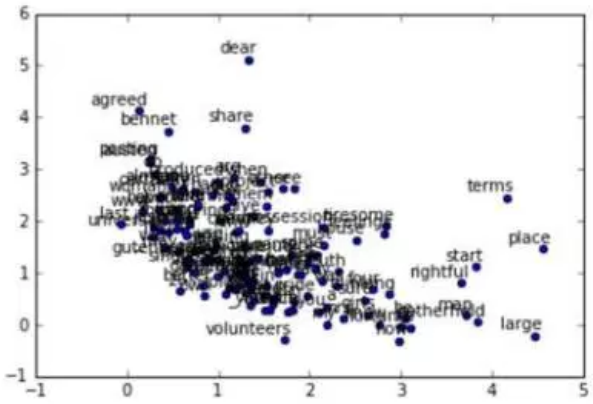
$$J_{\text{NEG}} = \log Q_\theta(D=1|w_t,h) + k \mathop{\mathbb{E}}_{\tilde{w}\sim P_{\text{noise}}} [\log Q_\theta(D=0|\tilde{w},h)]$$

Where we consider both positive and negative samples and try to maximise it, where we select k noise words from the negative samples. The function is scaled on the number k, not necessarily being the size of your whole vocabulary. In TensorFlow, this is called Noise-Contrastive Estimation (NCE) loss, where the single function would help you with the work: tf.nn.nce_loss().
(lol you will find that lots of keywords will lead you to a paper in Google TF tutorial)

T-SNE

A helpful method for you to visualize your high-dimensional data. If you are familiar with PCA, given a high-d vector, we could

use 2-d or 3-d to represent it, which allows us to plot it in a proper low dimension. When the word embeddings are trained (it is usually a fixed-size set of vectors), we could use T-SNE to visualize the word clusters.



(A trained word embedding visualization from the Pride and Prejudice)

The trained embedding is ready to use for RNNs.

Hope this blog will help you understanding the TF tutorial in Word Representation.
For the implementation part, I will update part 2 as a new blog.

Related

TensorFlow 04 : Implement a LeNet-5-like NN to classify notMNIST Images
In "Algorithm"

Deep Learning 16: Understanding Capsule Nets
In "Deep Learning"

NLP 05: From Word2vec to Doc2vec: a simple example with Gensim
In "Algorithm"

Author: Irene

Keep calm and update blog.

VIEW ALL POSTS

Previous Post

NLP 04: Log-Linear Models for Tagging Task (Python)

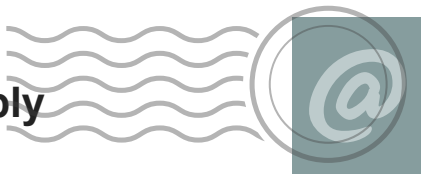
Next Post

Deep Learning 11: Energy-Based Learning (1)–What is EBL?

One thought on “TensorFlow 06: Word Embeddings (1)”

Pingback: TensorFlow 07: Word Embeddings (2) – Loading Pre-trained Vectors – Café, bonne nuit

Leave a Reply



Enter your comment here...