<https://www.quora.com/How-does-fastText-output-a-vector-for-a-word-that-is-not-in-the-pre-trained-model>

Currently i'm working on a sentiment analysis research project using LSTM networks.

As the input I convert sentences into set of vectors using word2vec.

And there are some well pretrained word vectors like Google word2vec.

My problem is, is there are any advantages of using custom trained word2vecs(train using a dataset which related to our domain, such as user reviews of electronic items) over pretrained ones.

Whats the best option

1. use a pretrained word2vec
2. Train our own word2vec using a dataset related to the domain

Ans :-

2down voteaccepted

The entire philosophy of Distributed Word Representations makes use of the fact that a word is understood by the context it has . When we say context , we mean the words that come in the neighborhood of a particular word . Now context in natural language is a tricky thing . As an example . the words ***open*** and ***create*** are two words that are not so similar semantically . But think of the scenario of a Bank related corpus , you will frequently see statements like

How do I open a new account

&

How do I create a new account

In the context of such a document , ***open*** and ***create*** become similar words. That is why context specific word vectors are needed.

Now , when it comes to which is the better option between the two

1. Use pre-trained vectors
2. Use custom vectors ,

it depends on how much data do you have for your custom use case . If you have enough data then it's always safe/better to go for a custom vectorization as it will be very specific to the context the corpus has . But in all other cases , you can use the pre trained embeddings. These generalize quite well on a variety of docs.

Ans:

2down voteaccepted

The advantage of using pre-trained vectors is being able to inject knowledge from a larger corpus than you might have access to: [word2vec](https://code.google.com/archive/p/word2vec/) has a vocabulary of 3 million words and phrases trained on the google news dataset comprising ~100 billion tokens, and there's no cost to you in training time.

How to handle oov words?

Random initializing or zero… mean

* Which off-the-shelf model should be use? Depending on your data, it**is possible that all of them are not useful for your domain specific data**.
* Should we train word embedding layer base on your data? According to my experience, if you deal with **domain specific text and most of your word cannot be found from off-the-shelf model**, you may consider to build customize word embedding layer.
* Tensorboard picks first 100000 vectors due to browser resource concern. Recommend to **pick a small portion of vectors** by yourself.